Using CMake to build libSBML-5

This document describes the libSBML-5 CMake Build system.

CMake:

CMake is a cross platform build system that not only enable users to build libSBML but also to generate projects for commonly used IDEs such as Visual Studio, Eclipse, Xcode and CodeBlocks. CMake uses the concept of out-of-source builds. This makes it easier to have several builds with different options existing side-by-side.

CMake provides a graphical user interface (cmake-gui) that makes it easy to configure the build options for different IDEs as well as having a command line option that can be used in a similar fashion to 'make'.

Requirements:

CMake Version 2.8 or higher, freely available from: www.cmake.org.

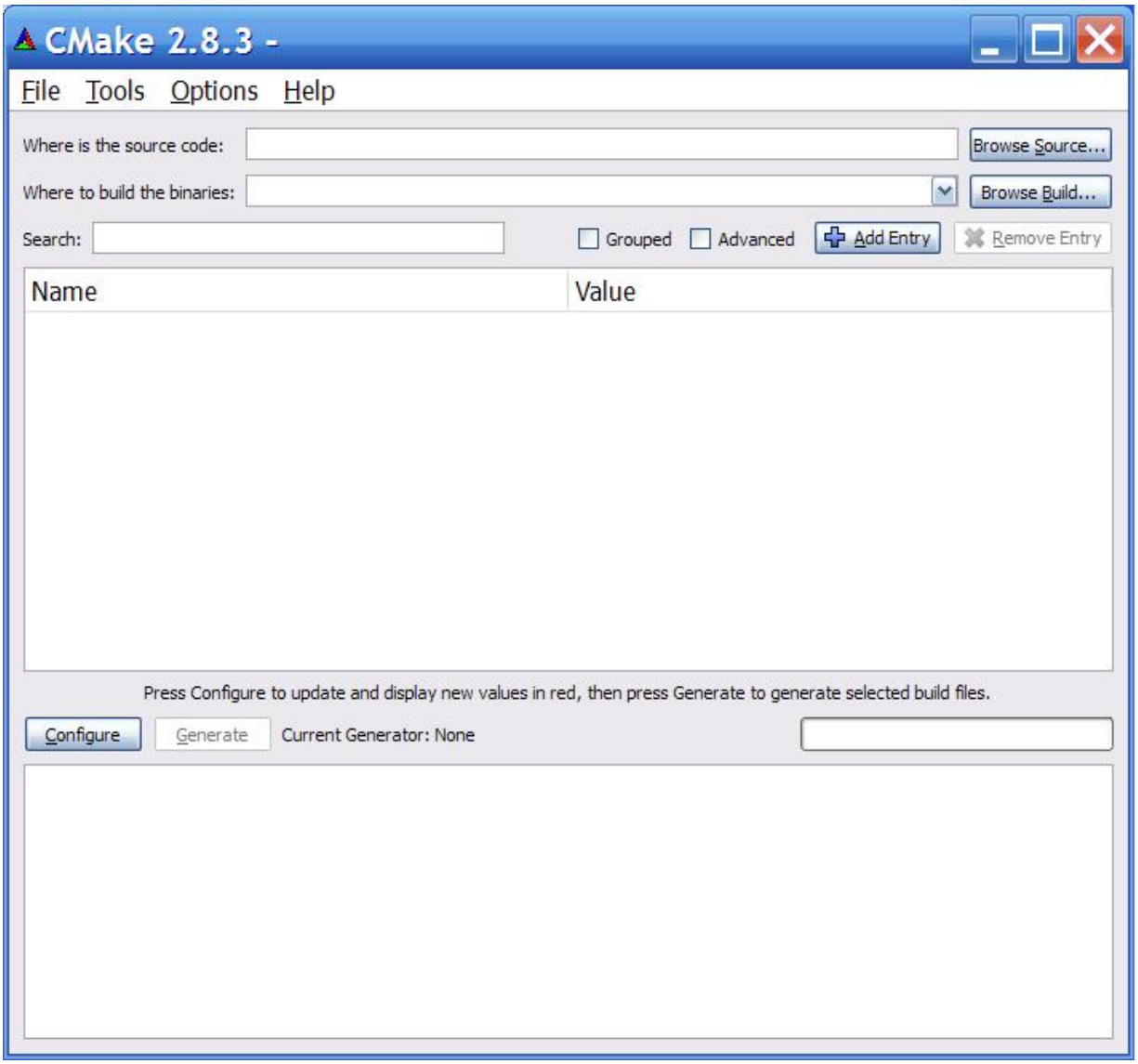
Note:

The existing libSBML Makefiles will still work as in previous versions of libSBML. However, we do anticipate replacing the build system with CMake in the future.

Contents

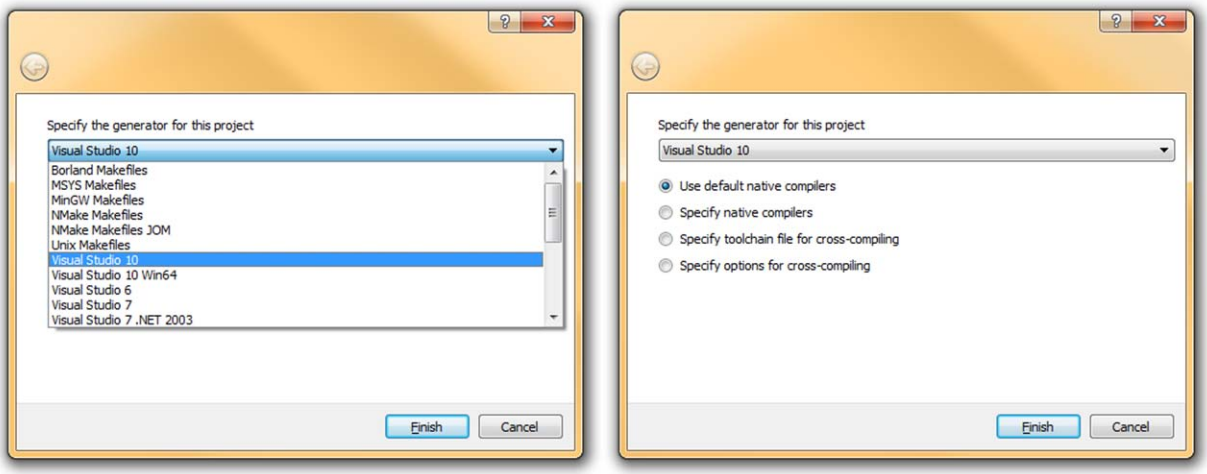
1. Using the CMake GUI

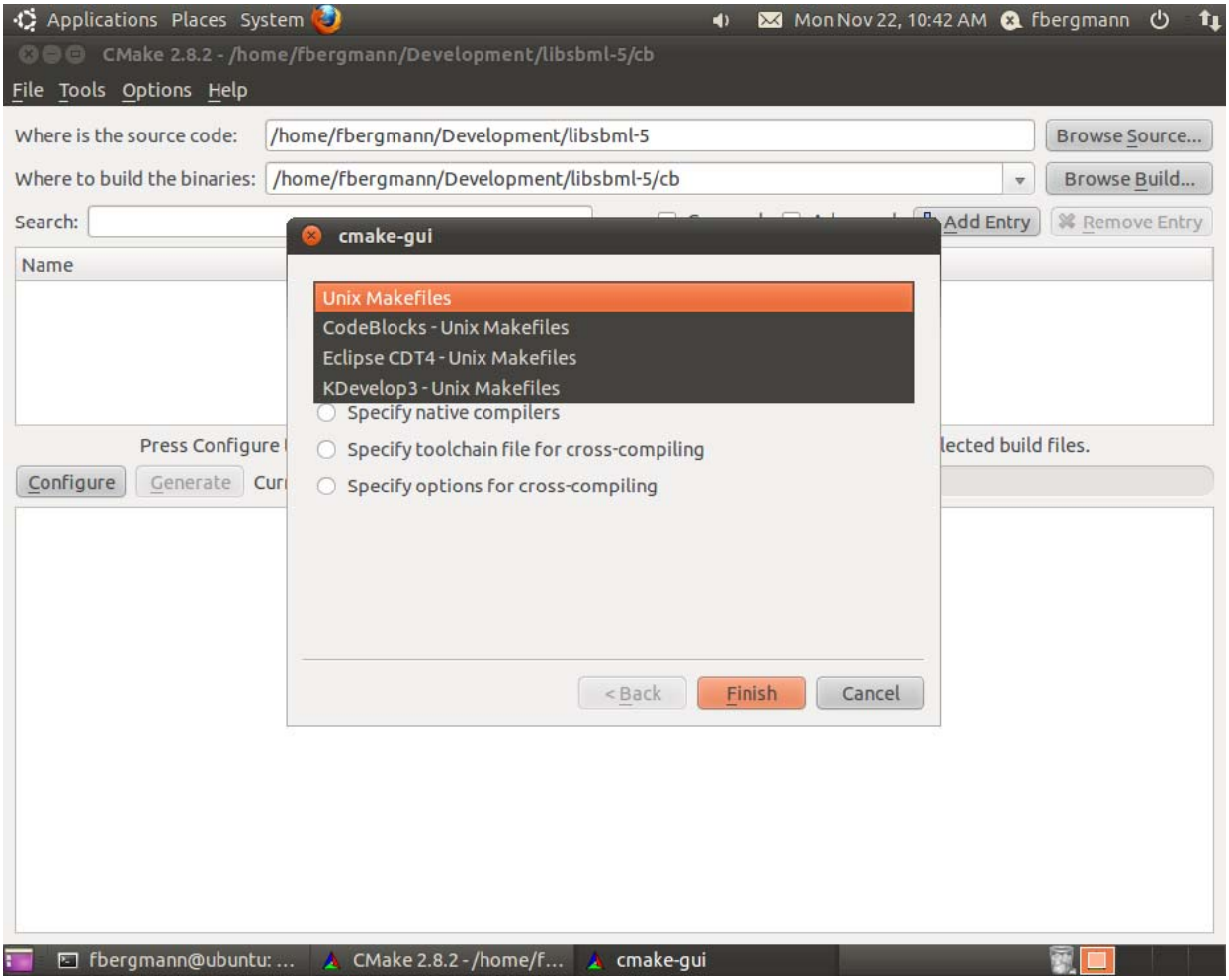
1.1 Start screen



Browse or enter the libsbml-5 source directory and a build directory; where all binaries and configuration files will be placed (if this directory does not exist it will be created).

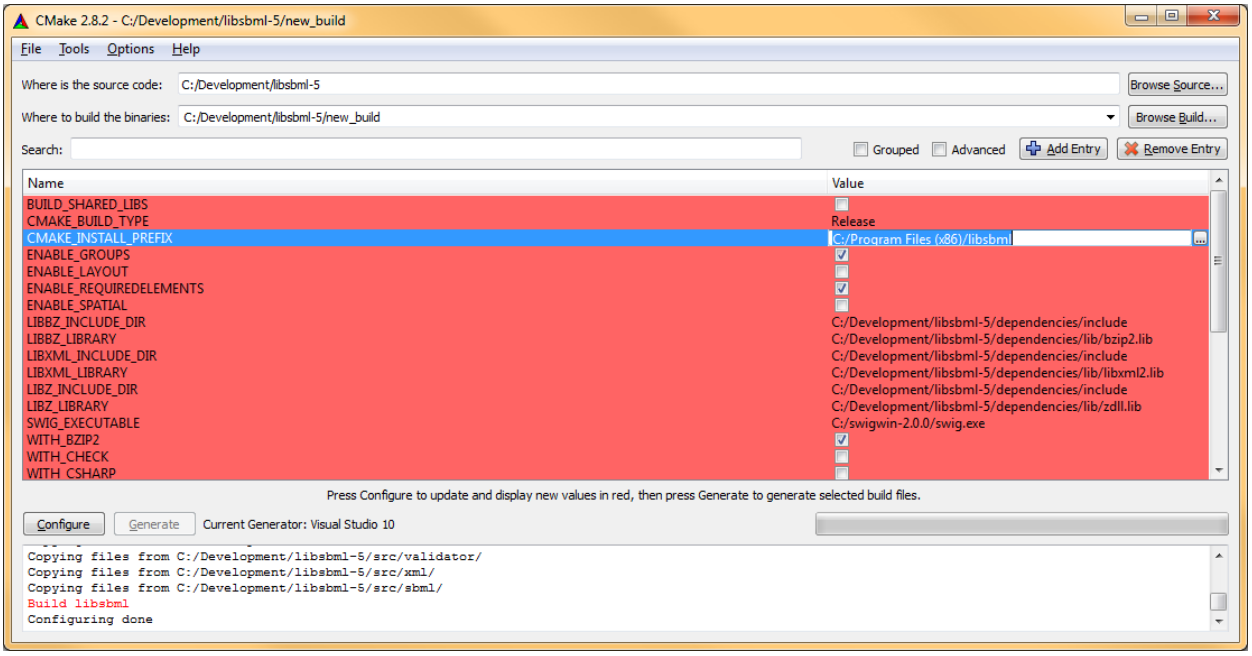
Click on “Configure”. This brings up a dialog box listing the possible build options. This list will vary depending on the OS you are working on. Here we show an example that selects Visual Studio 2010 on a Windows OS whereas below shows the options on Ubuntu 10.10.





1.2 Configuration

Click “Finish”. CMake will try and find all dependencies for the default options and displays all the newly added options in red.



Click “Configure” and CMake will process the configuration until it has all the necessary information. Note the all options for configuring libSBML, L3 packages and language bindings are listed and may be selected/deselected as required. It may be necessary to click “Configure” a number of times until all options are no longer displayed in red and the “Generate” button becomes enabled.

Click “Generate” and CMake will create the appropriate files for the build and configuration requested. These will be available in the directory specified for building the binaries.

2. Using CMake from the command line

In the top level libSBML-5 directory create a new directory in which the build files will be created. Change to this directory and use the command

cmake ..

This will build libSBML-5 with the default options.

Further options can be supplied to the command line using –D. Some examples are

-DENABLE_LAYOUT:BOOL="1" -DLIBXML_INCLUDE_DIR:PATH="/usr/include/libxml2"

3. Using CMake with Visual Studio

Once a build environment has been selected CMake will try and find all dependencies for the default options. On Windows this can cause issues and so we recommend that you use the dependencies available from:

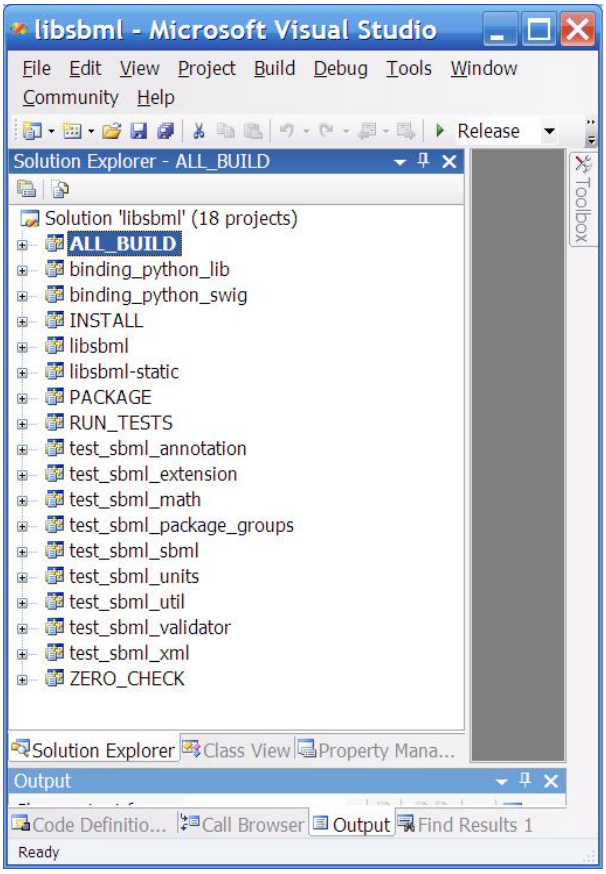
https://sourceforge.net/projects/sbml/files/libsbml/win-dependencies

Extract this file into the libsbml-5 source tree and all the default libraries as needed by Visual Studio will be found. Note CMake looks for these dependencies in a folder named 'dependencies' directly below the libsbml-5 root directory.

Once CMake has configured and generated the projects there is a .sln file available in the build directory. This will open the Solution in whichever version of MSVC was specified.

3.1 MSVC Projects

The MSVC Solution will contain a number of project files; depending on the configuration used.



The ALL_BUILD target builds all the libSBML project files; i.e. all projects except ones that involve installation/packaging/testing.

The PACKAGE target allows creating binary installers for the current platform. If NSIS (the Nullsoft scriptable installation system, http://nsis.sf.net) is available a windows installer is created. At present if NSIS is not available this process will fail. However a ZIP archive can be generated from the command line (see Section 6).

The INSTALL target will install the binaries to the directory specified in the CMAKE_INSTALL_PREFIX option.

The ZERO_CHECK target is merely a project used to check that all CMake files are up to date.

The RUN_TESTS target will test all the libraries built. This project is built if WITH_CHECK is selected at the configuration stage. It should be noted that the libcheck library is incompatible with MSVC7 and thus a later version is required to use the check facility. Note that these checks will fail if windows is unable to locate the dynamic libraries or for language bindings if it is unable to locate both the binding library and the native library. CMake sometimes adds a 'release' directory to the anticipated location of a library file. This issue will be addressed.

Other project files are named to indicate the intended target. Some examples include:

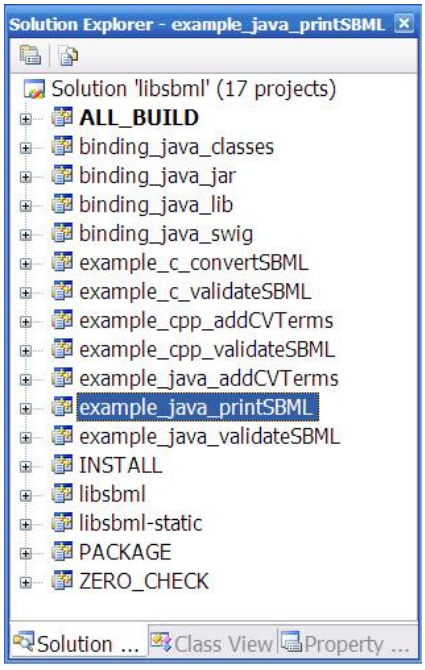
binding_python_lib - building the _libsbml.pyd python library

binding_java_classes – building the java class jar

example_c_convertSBML – building the convertSBML example in C

example_java_addCVTerms.java – building the addCVTerms example in java

test_sbml_math – building the tests on the src/math directory code



4. Using CMake with cygwin

The standard setup of cygwin includes 'cmake' as an option and thus this should be installed within the cygwin environment. CMake can then be used within cygwin as a command line tool (see Section 2).

5. Building the MATLAB binding

Prior to using a particular compiler to build the MATLAB binding it is necessary to run a set up script within MATLAB. At the command prompt type

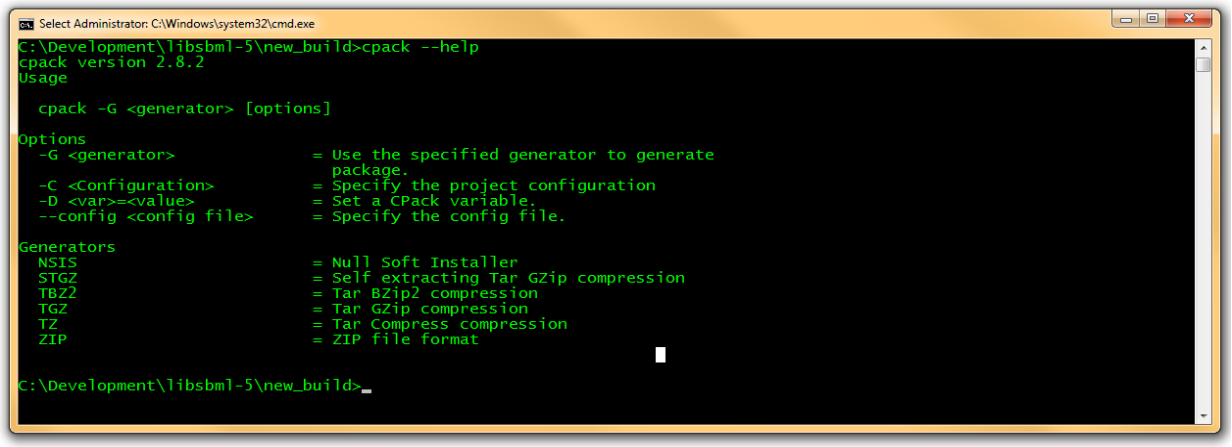
> mex -set-up

MATLAB will ask if you wish it to detect available compilers. It is best to allow this and then chose the compiler you intend to use from the list supplied.

This step need only be done once, unless the configuration changes the Generator being used.

6. Creating installer packages

As mentioned in Section 3.1 it is possible to create a windows installer package. On other platforms Debian packages, RPMs, DMGs or package maker files are also available. Navigating to the build directory and typing "cpack –help" displays the available package systems on the build machine as shown.



The command "cpack –G ZIP" would then create a zip file.