# Road Runner Project

## Table of Contents

[1 Road Runner Project 1](#_Toc313356865)

[1.1 Table of Contents 1](#_Toc313356866)

[2 Steps and info for library installs and builds. 2](#_Toc313356867)

[2.1 Compiling libsbml 2](#_Toc313356868)

[2.1.1 Generate Visual Studio project files using CMake 2](#_Toc313356869)

[2.1.2 Compile using Visual Studio 3](#_Toc313356870)

[2.2 Compiling NOMLib 4](#_Toc313356871)

[2.3 Compiling SBW core 4](#_Toc313356872)

[2.4 Compiling libStruct 4](#_Toc313356873)

[2.5 Compiling RoadRunner 5](#_Toc313356874)

Last time edited :

External svn revisions:

libsbml revision: 15157  
 libstruct: 16   
 roadrunner: 79  
 snowburst: 72  
 sbw\_core: 272

# Steps and info for library installs and builds.

A public repository for files and source code related to the project is available at:

<https://code.google.com/p/roadrunnerwork/>

This repository also contain required svn externals to third-party libraries, i.e. libsbml, NOMLib, SBW Core, libstruct and roadrunner. These externals are tied to a fixed revision (see page 1). To update to a later revision, the properties of the external need to be modified.

In the following, when referring to a checked out working copy, its root folder is denoted: *svn\_root*.

Default output of libraries and DLL's are set in CMake to be c:/sbw\_libs, i.e.

CMAKE\_INSTALL\_PREFIX **=** c**:**/sbw\_lib

## Compiling libsbml

The libsbml library code is located at svn\_root/Externals/libsmbl.

The repository address is https://sbml.svn.sourceforge.net/svnroot/sbml/trunk/libsbml.

Short instructions on how to compile libsmbl is in the document:

svn\_root/Docs/libSBMLBuildWindows.docx

which is an excerpt from the main libsbml documents. One important step mentioned in that document is to download win-dependencies. In the following only win32 (not x64) is being considered.

### Generate Visual Studio project files using CMake

The CMake GUI is used to generate solution files for Visual Studio.

When running CMake, the following settings were used:  
 Generator for project: VisualStudio10 with option "Use default native compilers".

The following environment flags in CMAKE were modified to:

CMAKE\_INSTALL\_LIBDIR **=** C**:/sbw\_libs/libsbml/**lib

CMAKE\_INSTALL\_PREFIX **=** C**:/sbw\_libs/libsbml**

For the current release of libsbml, a python interpreter need to be available on the system (not mentioned in their build docs). An additional flag has to be added in the CMake UI :

PYTHON\_EXECUTABLE = 'path and filename of python interpreter'

See <https://sourceforge.net/tracker/?func=detail&atid=942737&aid=3441615&group_id=71971> for more info.

Clicking the "generate" button in CMake UI generates VisualStudio solution and project files in the folder

svn\_root**/**Externals**/**libsbml**/**build

### Compile using Visual Studio

The solution file **libsbml.sln** is opened in Visual Studio and the target ALL\_BUILD project can be built. After finishing successfully, the INSTALL target is built, which copies all relevant libs and dll files, using the folders defined above by CMAKE\_INSTALL\_LIBDIR and CMAKE\_INSTALL\_PREFIX, i.e. C**:/sbw\_libs/libsbml**.

## Compiling NOMLib

NOMLib is part of Snowburst, and is "*a C API to libSBML that implements the original NOM API used in SBW"*, from documentation at <http://snowburst.googlecode.com/svn/NOMLib/help/html/index.html>

Snowburst is found in the externals folder, i.e. svn\_root/Externals/snowburst. Actual repository address is: https://snowburst.googlecode.com/svn.

NOMLib is built using Visual Studio by the solution file: NOMLib.sln, found in snowburst/NOMLib

After compiling, NOMLib binaries and link libraries are found in:  
svn\_root\Externals\snowburst\NOMLib\bin\Debug

Currently, NOMLib seem to be compiled and linked against a "older(?)" version of the sbml library, contained within the snowburst's repository. There are multiple, non identical, copies of various dll's in the NOMLib bin\Debug and the libsbml\bin folder which could cause trouble(?).

For now, a NOMLib CMake project was added to this projects repository. This project links against the libsbml files generated above (section 2.1). The resulting library, .lib, .dll and corresponding header file(s) is installed in (by setting CMAKE\_INSTALL\_PREFIX) to c:/sbw\_libs/NOMLib

## Compiling SBW core

The SBW core is found in svn\_root/Externals/sbw\_core with actual repository address: http://sbw.svn.sourceforge.net/svnroot/sbw/trunk/core

To build, open the solution file SBW-vs2010.sln found in folder: svn\_root\Externals\sbw\_core\VisualStudio

This build creates a SBWD.dll in .\sbw\_core\bin and import library SBWD.lib in sbw\_core\lib, needed for building libstruct, see below.

## Compiling libStruct

libStruct is a C/C++ library. It is found in svn\_root/Externals/libStruct with actual repository address https://libstruct.svn.sourceforge.net/svnroot/libstruct/

In the following, instructions found in the Readme.pdf file in the root folder of libStruct was consulted. Main strategy is to use CMake to generate project and solution files for Visual Studio.

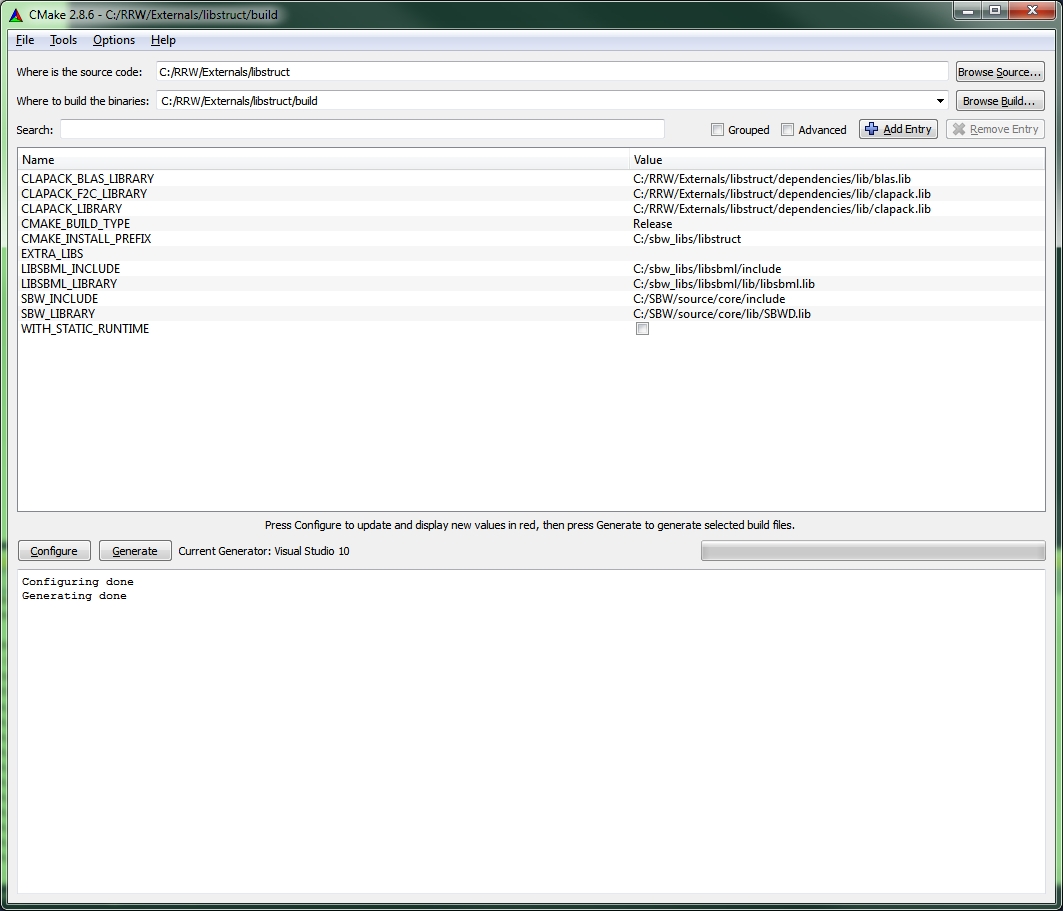
Running CMake GUI and pointing it to libStructs root folder, and creating a folder, build, in the same for the CMake output files, a few errors occurs, which need attention:

CMake will complain that the following flags are undefined:

CLAPACK\_F2C\_LIBRARY  
 LIBSBML\_INCLUDE  
 LIBSBML\_LIBRARY  
 SBW\_INCLUDE  
 SBW\_LIBRARY

To fix the above;  
 CLAPACK\_F2C\_LIBRARY is pointed to ./libstruct/dependencies/lib/clapack.lib  
LIBSBML\_INCLUDE is set to SBML includes, i.e. C:/sbw\_libs/libsbml/include  
LIBSBML\_LIBRARY is set to C:/sbw\_libs/libsbml/lib/libsbml.lib  
SBW\_INCLUDE is set to svn\_root\Externals\sbw\_core\include  
SBW\_LIBRARY is set to svn\_root\Externals\sbw\_core\lib\SBWD.lib

The visual studio solution file, libstruct/build/LibStructural.sln is opened in VisualStudio and target ALL\_BUILD can be built. If finished, successfully, the INSTALL target is executed, which installs the dll's and libraries to the folder specified in CMakes CMAKE\_INSTALL\_PREFIX.



## Compiling RoadRunner

By running the BuildSBW.exe, various code bases are checked out automatically. The UI allows to build C++ and C# libraries.

There was several build errors due to locked svn folders. Manually cleaning and rebuilding resulted eventually in a successful build of all components.

Roadrunner was also compiled successfully from the Visual Studio IDE, by opening the solution file in SBW/source/RoadRunner.

In addition, Road Runner was also compiled 'outside' the BuildSBW.exe framework, i.e. using the solution file in RRW/Externals/roadrunner. However, the repository have 'bad' DLL's (libsbmlcsP.dll) in the folder roadrunner/3rdParty, causing the build to fail. Deleting them from the folder allows the solution to build successfully.

## Conversions

Status Colors Green - in progress Grey - on hold White - Not Started Blue - To be tested

|  |  |  |  |
| --- | --- | --- | --- |
| **C# File Name (.cs)** | **C++ File Name + (.h and .cpp)** | **Delphi (.pas)** | **Notes** |
| PendingAssignment | rrPendingAssignment |  |  |
| RoadRunner.FreqRes |  |  |  |
| supportCode | rrSupportCode |  | Rewriting all functions to "regular" ones, i.e. non static. |
| ModelState | rrModelState |  |  |
| MathKGI | rrMath | rrMath | No 'params' c# yet converted  Factorial should be 'int' ? |
| RoadRunner.MCA |  |  |  |
| ModelGenerator |  |  |  |
| RoadRunner |  |  |  |
| TComputeEventAssignmentDelegate | rrTComputeEventAssignmentDelegate |  |  |
| TEventAssignmentDelegate |  |  |  |
| TEventDelayDelegate |  |  |  |
| TEventDelegate |  |  |  |
| TParameterType |  |  |  |
| TPerformEventAssignmentDelegate | rrTPerformEventAssignmentDelegate |  |  |
| TVariableType |  |  |  |
| IModel |  |  |  |
|  |  |  |  |

### Conversion notes

General conversions

|  |  |  |
| --- | --- | --- |
| C# | C++ | notes |
| convertSpeciesToY(string str) | convertSpeciesToY(const string& str) | Pass by reference in c++ |
| private: static ModelGenerator \_instance;  private: vector<int> \_LocalParameterDimensions;  private: string \_ModelName;  private: int \_NumBoundarySpecies;  private: int \_NumCompartments; | private:  static ModelGenerator \_instance;  vector<int> \_LocalParameterDimensions;  string \_ModelName;  int \_NumBoundarySpecies;  int \_NumCompartments; | Have one private, protected, public section in c++ header . |

|  |  |  |
| --- | --- | --- |
| C# file | Notes | Depends on |
| MathKGI.cs | C# 'params' functions not yet translated - need to use va\_list and , ... syntax for those  Factorial - should only deal with integers? |  |
| PendingAssignment.cs |  | TComputeEventAssignmentDelegate.cs  TPerformEventAssignmentDelegate.cs |
| TComputeEventAssignmentDelegate.cs | Function pointer |  |
| TPerformEventAssignmentDelegate.cs | Function pointer |  |
| ModelState.cs |  | IModel.cs |
| IModel.cs | Interface class | TEventDelayDelegate.cs  TEventAssignmentDelegate |
| RoadRunner.cs | Need CvodeInterface.cs to be converted. | CvodeInterface.cs |
| ModelGenerator.cs | Huge.... |  |

## Tests

Various test applications are located in Testing folder. These test are there to test various concepts, classes throughout the conversion process.

## Convert Small Files

# Appendix

### Abbreviations and other entities

cvode : a solver for stiff and nonstiff ordinary differential equation (ODE) systems