openPASS / World_OSI

Setup guide

Date	06.09.2019
Author	Reinhard Biegel

Contents

1 Introduction

The World_OSI provides an implementation of the WorldInterface using Open Simulation Interface (OSI) as a “backend storage” [1]. OSI already provides data structures for representing various objects in traffic simulation environments. As these objects are specified rather sensor centric, the OSI source currently needs some additions to satisfy the needs of the World_OSI module (see OSI WorldInterface Pull Request [2]).

To allow unit-testing of the implementation, an additional layer has been introduced on top of OSI, whose classes represent the objects needed in the simulator’s world (WorldData). The OSI objects are instantiated there. An architectural diagram showing WorldData and its interactions is currently not available.

Another benefit of the additional WorldData layer is keeping the area of contact to the data backend small, to allow easy adaption if OSI switches from protobuf to another storage technology (open discussion).

The current use of protobuf provides easy serialization and deserialization of all stored data out-of-the-box.

2 Prerequisites

You’ll find the openPASS sources in Eclipse Gerrit [9]. Please note to check out the **intech** branch, where the OSI changes will be located for now. Of course, associated changes on Gerrit are also an option. For the OSI world demonstration an additional project file `OpenPASS_OSI_UseCase.pro` has been added, which can be opened in QtCreator IDE, or directly be built using `qmake` on the command line.

It is recommended to use a **short path for the source code checkout**, due to limitations in the MinGW/Windows setup. If you ever come across compilation errors regarding files not being found, you might have run into problems with path length restrictions.

Before being able to compile and run the World_OSI, make sure to have all dependencies installed.

World_OSI has been developed on Linux x86_64 using Qt 5.12.3, gcc 7.1.0, 7.3.0 and 8.1.0. On Windows systems a MinGW environment with the same compiler versions has been used. There are no known issues regarding different gcc versions.

To summarize, openPASS with World_OSI has these dependencies:

- MinGW gcc environment (Windows, testet gcc versions are 7.1.0, 7.3.0 and 8.1.0, 64-bit)
- Qt 5.12.3 (incl. QtCreator IDE and qmake build system)
- Open Simulation Interface (OSI)

- Google Protobuf 3.6.1
- Cmake 3.9.4
- boost library 1.63
    - algorithm
    - math
    - geometry
- Google Test / Google Mock 1.8.1

3 Dependency installation guide

The World_OSI module introduces additional third-party dependency on the OSI library, and as a consequence on Google protobuffers. Furthermore, the boost (geometry) library is required to compile and run World_OSI [5]. Other dependencies have to be set up for the build environment itself.

3.1 MinGW

An installation for 64-bit MinGW environments is provided at [8]. You can download a generic installer binary, which will let you select different Versions of MinGW. Latest tested setup is `x86_64-8.1.0-posix-seh-rt_v6-rev0`.

3.2 CMake

For compilation of protobuf and OSI, the CMake build system is required. Please download and install a binary distribution of CMake [4]. Version 3.9.4 has been used during testing. Please feel free to try a more recent version, but note that issues with newer versions have already been reported.

3.3 Protobuf

If not already installed on the system, the protobuf library and headers have to be built prior to being able to compile OSI. Protobuf sources are located at [3]. This guide will give some instructions to compile version 3.6.1, but any other version should be compatible (including 2.x).

As most users seem to work on Windows environments, the following instructions will target these systems.

1. Download `protobuf-cpp-3.6.1.zip` and extract the archive. We will use `C:\OpenPASS\thirdParty\sources` as target directory.

2. Open a command line with MinGW environment at the extraction directory

3. Create the build directory

```
> cd cmake
> mkdir build
> cd build
```

4. Run CMake (tests are disabled here due to some compiler warnings being treated as errors, may vary with compiler version)

```
> cmake -G "MinGW Makefiles" -DCMAKE_BUILD_TYPE=Release
  -DCMAKE_INSTALL_PREFIX=C:/OpenPASS/thirdParty
  -Dprotobuf_BUILD_SHARED_LIBS=ON
  -Dprotobuf_BUILD_TESTS=OFF ..
```

5. Compilation

```
> mingw32-make -j3
```

6. Installation

```
> mingw32-make install
```

7. Final steps
   a. Copy the MinGW runtime libraries from the MinGW installation directory to the `thirdParty/bin` directory (`libgcc_s_seh-1.dll`, `libwinpthread-1.dll` and `libstdc++6.dll`). This allows running the protobuf compiler (`protoc.exe`) outside of a full MinGW environment.
   b. Copy the `libprotobuf.dll` from `bin` to `lib` subfolder

3.4 Open Simulation Interface

The OSI project is hosted on Github. Please check out the source of the mentioned **pull request at [2].** Compile OSI using the following commands executed from the source directory.

1. Create build directory

```
> mkdir build
> cd build
```

2. Run CMake. As I could not make CMake find protobuf in the custom installation directory, we specify the paths manually here.

```
> cmake -G "MinGW Makefiles" -DCMAKE_BUILD_TYPE=Release
  -DCMAKE_INSTALL_PREFIX=C:/OpenPASS/thirdParty
  -DProtobuf_INCLUDE_DIR=C:/OpenPASS/thirdParty/include
  -DProtobuf_PROTOC_EXECUTABLE=C:/OpenPASS/thirdParty/bin/protoc.exe
  -DProtobuf_LIBRARIES=C:/OpenPASS/thirdParty/lib ..
```

3. Adding linker flags for protobuf

Edit the file `C:\OpenPASS\thirdParty\source\open-simulation-interface\build\CMakeFiles\open_simulation_interface.dir\linklibs.rsp`

Add `"-LC:/OpenPASS/thirdParty/lib -lprotobuf"` to the end of the line.

If anybody knows how to avoid this step, please drop a note.

4. Compilation

   ```
   > mingw32-make -j3
   ```

5. Installation

   ```
   > mingw32-make install
   ```

6. Final steps
   a. Include path fix: rename `thirdParty/include/osi3` to `osi`
   b. Copy `libopen_simulation_interface.dll` from `lib/osi3` subfolder to `lib`

The OSI class documentation is part of the source code and can be compiled using Doxygen. Instructions are located in the OSI `Readme.md`. A Pre-compiled version is located here: https://opensimulationinterface.github.io/open-simulation-interface/index.html. Note that this version of the documentation doesn't include the extensions from the OSI WorldInterface pull request.

3.5 boost

Boost consists of multiple libraries, many of them being header-only [5]. At the moment, World_OSI only makes use of the `algorithm`, `math` and `geometry` header-only-libraries.

1. Download and extract boost from [6]. Boost version 1.63 was used during development of World_OSI. Newer versions should be compatible, but have not been tested yet.

2. Extract downloaded archive to a temporary directory

3. Copy the `boost` subfolder to your thirdParty/include directory

3.6 Google Test and Google Mock

For unit-testing, World_OSI and other modules make use of the gtest/gmock framework. Installation requires `cmake`, just like `protobuf`, which was mentioned above.

1. Download a release from [7].

Version 1.8.1 has been used during development.

2. Extract the archive to a temporary directory

3. Open a MinGW shell at the extraction directory

4. Create the build directory

```
> mkdir build
> cd build
```

5. Run CMake

```
> cmake -G "MinGW Makefiles"
  -DCMAKE_INSTALL_PREFIX=C:/OpenPASS/thirdParty ..
```

6. Compilation

```
> mingw32-make
```

7. Installation

```
> mingw32-make install
```

4 Compiling World_OSI

Please adapt the variable `EXTRA_LIB_PATH` and `EXTRA_INCLUDE_PATH` to your needs. These are set in `OpenPass_Source_Code/defaults.pri`. Using the paths from this guide, the required values would have to be `C:/OpenPASS/thirdParty/lib` and `C:/OpenPASS/thirdParty/include` respectively. It is also possible to pass these variables on the `qmake` command line, which will then override the default values.

5 Simulation

The proposed demo scenario can be simulated using the configuration files from `OpenPass_Source_Code/openPASS_Resource/OpenPass_OSI_UseCase`.

NOTE:
Remember to copy the required third party libraries (OSI, protobuf, Qt, MinGW) to the directory, where the OpenPassSlave.exe is located. Dependency locations during runtime are (mostly) independent from compile-time locations. Alternatively, you can add a directory containing these libraries to the PATH environment variable.

6 References

[1] https://github.com/OpenSimulationInterface/open-simulation-interface

[2] https://github.com/OpenSimulationInterface/open-simulation-interface/pull/244

[3] https://github.com/protocolbuffers/protobuf/releases

[4] https://cmake.org/files/v3.9/

[5] https://www.boost.org/

[6] https://sourceforge.net/projects/boost/files/boost/

[7] https://github.com/google/googletest

[8] https://sourceforge.net/projects/mingw-w64/

[9] https://git.eclipse.org/r/#/admin/projects/simopenpass/simopenpass