# Building SOEM

## Contents

- Prerequisites
- Installation
- Building SOEM and samples
- Using SOEM in your application

This tutorial shows how to build SOEM on your Windows or Ubuntu Linux PC.

## Prerequisites

This tutorial requires a Windows or Ubuntu Linux system with the following tools installed. The next chapter explains how to install tools that are not already on your system.

Ubuntu Linux **Windows**

- CMake 3.28 or later
- Visual Studio 2022 Build Tools and Native Desktop workload

## Installation

Follow the instructions below to install the required tools for building CMake. You will need to have administrative privileges to be able to install software on your system.

Ubuntu Linux **Windows**

We will be using Chocolatey for package management, as it greatly simplifies the installation of the build tools.

Start by installing Chocolatey if you don't have it already. Save the Chocolatey install script shown below as `installChocolatey.cmd`:

```
@echo off

SET DIR=%~dp0%

::download install.ps1
%systemroot%\System32\WindowsPowerShell\v1.0\powershell.exe -NoProfile -Executi
::run installer
%systemroot%\System32\WindowsPowerShell\v1.0\powershell.exe -NoProfile -Executi
```

Next open a Command Prompt as Administrator (as administrative rights are required to install software):

- Press `Win` + `R` to open the Run dialog.
- Type `cmd` and press `Shift` + `Ctrl` + `Enter` to open Command Prompt as Administrator.

Use the `cd` command to go to the folder where you saved the script and then run it. For example:

```
cd C:\Path\To\Script
installChocolatey.cmd
```

This should download and install Chocolatey, Once it finishes, restart the Command Prompt to start using Chocolatey.

Then install CMake, Python and the Visual Studio Build tools:

```
choco install -y cmake
choco install -y visualstudio2022buildtools
choco install -y visualstudio2022-workload-vctools
```

You can skip tools that are already installed on your system. Note that installing the Visual Studio tools will take a while.

# Building SOEM and samples

Run the following to configure and build SOEM.

Ubuntu Linux **Windows**

```
cmake --preset default -G "Visual Studio 17 2022"
cmake --build --preset default
```

When the build finishes, the samples will be in `build/default/bin`.

# Using SOEM in your application

There are two ways to include the SOEM library in your application.

## Option 1: Include SOEM as a subdirectory

You can add the SOEM sources directly as a subdirectory in your project structure:

```
├── CMakeLists.txt
├── main.c
└── soem
    ├── CMakeLists.txt
    └── [...]
```

Your top-level `CMakeLists.txt` should look like this:

```
cmake_minimum_required(VERSION 3.28)
project(APP VERSION 1.0.1)

add_subdirectory(soem)
add_executable(app main.c)
target_link_libraries(app soem)
```

## Option 2: Build and install soem separately

Alternatively, you can build and install SOEM separately. In the SOEM source tree, execute the following commands:

```
cmake --preset default -DCMAKE_INSTALL_DIRECTORY=/path/to/install
cmake --build --preset default --target install
```

Your project structure would then look like this:

```
├── CMakeLists.txt
└── main.c
```

Your top-level `CMakeLists.txt` in this case should be updated as follows:

```
list(APPEND CMAKE_PREFIX_PATH "/path/to/install/cmake")
cmake_minimum_required(VERSION 3.28)
project(APP VERSION 1.0.1)

find_package(soem REQUIRED)
add_executable(app main.c)
target_link_libraries(app soem)
```

Note that `CMAKE_PREFIX_PATH` must include the `cmake` folder of the SOEM install tree. You can also set the prefix path from the CMake command line or via a CMake preset.

## Summary

Choose the method that best suits your needs. Including SOEM as a subdirectory is simpler for small projects, while separate installation may be useful to share SOEM among multiple projects.