

---

## S2 Geometry Library

### Overview

This is a package for manipulating geometric shapes. Unlike many geometry libraries, S2 is primarily designed to work with *spherical geometry*, i.e., shapes drawn on a sphere rather than on a planar 2D map. This makes it especially suitable for working with geographic data.

If you want to learn more about the library, start by reading the overview and quick start document, then read the introduction to the basic types.

S2 documentation can be found on [s2geometry.io](https://s2geometry.io).

### API/ABI Stability

Note that all releases are version 0.x, so there are no API or ABI stability guarantees. Starting with 1.0 we will adhere to SemVer.

The Python API is particularly unstable, and it is planned that the SWIGged API will be replaced by a pybind11 version with more Pythonic names and more complete functionality.

### Requirements for End Users

- CMake
- A C++ compiler with C++14 support, such as g++ >= 5
- Abseil >= LTS 20240116 (standard library extensions)
- OpenSSL (for its bignum library)
- googletest testing framework >= 1.10 (to build tests and example programs, optional)

On Ubuntu, all of these other than abseil can be installed via apt-get:

```
1 sudo apt-get install cmake googletest libssl-dev
```

Otherwise, you may need to install some from source.

Currently, Abseil must always be installed from source. See the use of `-DCMAKE_PREFIX_PATH` in the build instructions below. This is likely to change.

On macOS, use MacPorts or Homebrew. For MacPorts:

```
1 sudo port install cmake openssl
```

Do not install `gtest` from MacPorts; instead download release 1.10.0, unpack, and substitute

---

```
1 cmake -DGOOGLETEST_ROOT=/...absolute path to.../googletest-release
   -1.10.0 ..
```

in the build instructions below.

Thorough testing has only been done on Ubuntu 14.04.3 and macOS 10.12.

## Build and Install

You may either download the source as a ZIP archive, or clone the git repository.

### Via ZIP archive

Download ZIP file

```
1 cd [parent of directory where you want to put S2]
2 unzip [path to ZIP file]/s2geometry-master.zip
3 cd s2geometry-master
```

### Via git clone

```
1 cd [parent of directory where you want to put S2]
2 git clone https://github.com/google/s2geometry.git
3 cd s2geometry
```

## Building

First, install Abseil. It must be configured with `-DCMAKE_POSITION_INDEPENDENT_CODE=ON`. `s2geometry` must be configured to use the same C++ version that abseil uses. The easiest way to achieve this is to pass `-DCMAKE_CXX_STANDARD=14` (or `-DCMAKE_CXX_STANDARD=17`) to `cmake` when compiling both abseil and `s2geometry`.

From the appropriate directory depending on how you got the source:

```
1 mkdir build
2 cd build
3 # You can omit -DGOOGLETEST_ROOT to skip tests; see above for macOS.
4 # Use the same CMAKE_CXX_STANDARD value that was used with absl.
5 cmake -DGOOGLETEST_ROOT=/usr/src/googletest -DCMAKE_PREFIX_PATH=/path/
   to/absl/install -DCMAKE_CXX_STANDARD=14 ..
6 make -j $(nproc)
7 make test ARGS="-j$(nproc)" # If GOOGLETEST_ROOT specified above.
```

---

```
8 sudo make install
```

On macOS, `sysctl -n hw.logicalcpu` is the equivalent of `nproc`.

Disable building of shared libraries with `-DBUILD_SHARED_LIBS=OFF`.

Enable the python interface with `-DWITH_PYTHON=ON`.

If OpenSSL is installed in a non-standard location set `OPENSSL_ROOT_DIR` before running configure, for example on macOS:

```
1 OPENSSL_ROOT_DIR=/opt/homebrew/Cellar/openssl@3/3.1.0 cmake -
  DCMAKE_PREFIX_PATH=/opt/homebrew -DCMAKE_CXX_STANDARD=17
```

## Installing

From `build` subdirectory:

```
1 make install
```

Prefix it with `sudo` if needed:

```
1 sudo make install
```

*NOTE:* There is not `uninstall` target but `install_manifest.txt` may be helpfull.

All files will be installed at location specified in `CMACE_INSTALL_PREFIX` variable.

Several suffix variables used for some file groups:

Variable	Default	Description
<code>CMACE_INSTALL_INCLUDEDIR</code>	<code>include</code>	For header files
<code>CMACE_INSTALL_BINDIR</code>	<code>bin</code>	For executables and <code>*.dll</code> files on DLL-based platforms
<code>CMACE_INSTALL_LIBDIR</code>	<code>lib</code>	For library files ( <code>*.so</code> , <code>*.a</code> , <code>*.lib</code> etc)

If needed set this variables on command line as `cmake` arguments with `-D` prefix or edit from `build` subdirectory:

```
1 make edit_cache
```

For more info read: The CMake Cache.

---

## Python

If you want the Python interface, you need to run cmake using `-DWITH_PYTHON=ON`. You will also need to install the following dependencies:

- SWIG 4 (for Python support, optional)
- python3-dev (for Python support, optional)

which can be installed via

```
1 sudo apt-get install swig python3-dev
```

or on macOS:

```
1 sudo port install swig
```

Version 4.0 is required, but it should be easy to make it work 3.0 or probably even 2.0.

Python 3 is required.

## Creating wheels

First, make a virtual environment and install `cmake_build_extension` and `wheel` into it:

```
1 python3 -m venv venv
2 source venv/bin/activate
3 pip install cmake_build_extension wheel
```

Then build the wheel:

```
1 python setup.py bdist_wheel
```

The resulting wheel will be in the `dist` directory.

If OpenSSL is in a non-standard location make sure to set `OPENSSL_ROOT_DIR` when calling `setup.py`, see above for more information.

## Other S2 implementations

- Go (Approximately 40% complete.)
- Java
- Kotlin (Complete except binary serialization)

---

## **Disclaimer**

This is not an official Google product.