IRENE: AE9/AP9/SPM Radiation Environment Model

Build Instructions

Version 1.58.001

The IRENE (International Radiation Environment Near Earth): (AE9/AP9/SPM) model was developed by the Air Force Research Laboratory in partnership with MIT Lincoln Laboratory, Aerospace Corporation, Atmospheric and Environmental Research, Incorporated, Los Alamos National Laboratory and Boston College Institute for Scientific Research.

IRENE (AE9/AP9/SPM) development team: Wm. Robert Johnston[1] (PI), T. Paul O'Brien[2] (PI), Gregory Ginet[3] (PI), Stuart Huston[4], Tim Guild[2], Yi-Jiun Su[1], Christopher Roth[5], Rick Quinn[5], Michael Starks[1], Paul Whelan[5], Reiner Friedel[6], Chad Lindstrom[1], Steve Morley[6], and Dan Madden[7].

To contact the IRENE (AE9/AP9/SPM) development team, email ae9ap9@vdl.afrl.af.mil .

The IRENE (AE9/AP9/SPM) model and related information can be obtained from AFRL's Virtual Distributed Laboratory (VDL) website: https://www.vdl.afrl.af.mil/programs/ae9ap9

V1.00.002 release: 05 September 2012

V1.03.001 release: 26 September 2012

V1.04.001 release: 20 March 2013

V1.04.002 release: 20 June 2013

V1.05.001 release: 06 September 2013

V1.20.001 release: 31 July 2014

V1.20.002 release: 13 March 2015

V1.20.003 release: 15 April 2015

V1.20.004 release: 28 September 2015

V1.30.001 release: 25 January 2016

V1.35.001 release: 03 January 2017

V1.50.001 release: 01 December 2017

V1.57.004 release: 21 July 2022

V1.58.001 release: 04 March 2024

[1] Air Force Research Laboratory, Space Vehicles Directorate
[2] Aerospace Corporation
[3] MIT Lincoln Laboratory
[4] Confluence Analytics, Incorporated
[5] Atmospheric and Environmental Research, Incorporated
[6] Los Alamos National Laboratory
[7] Boston College Institute for Scientific Research

Build Instructions for IRENE (AE9/AP9/SPM) Radiation Environment Model Software, Version 1.58.001

The base distribution of the IRENE model package ships with pre-compiled single- and multi-threaded application binaries for the Windows platform only. To build the model and tools on other platforms, *the source code distribution must be obtained from the model development team*; send requests for the source code to: ae9ap9@vdl.afrl.af.mil .

IRENE (AE9/AP9/SPM) uses a CMake-based build process. CMake is a cross-platform make tool that supports most major operating systems and compilers. See https://cmake.org/documentation/ for more information. In theory, it should be possible to build and run the software in any CMake-supported environment that also supports all required third-party library dependencies.

Version 1.58.001 of the IRENE software has been tested under Windows10 (64-bit only) and under CentOS 7.x, CentOS/Rocky 8/9.x and Ubuntu 22.04 versions of the Linux operating system (64-bit). The software may be able to be built and run on other operating systems and versions of these operating systems, but formally verified on only those listed here.
The build process has significantly changed since the v1.50 release, with improved automation.

Third-Party Dependencies
Several third-party libraries are used with the IRENE (AE9/AP9/SPM) software. The IRENE package distribution does not ship with these libraries; they must be installed before building the software
➔The use of '*anaconda3*' packages for installing/fulfilling these libraries is *specifically NOT supported*.

Linux
Below is the list of system package names that are required to be installed on a CentOS 7.x system for a successful build of the IRENE software. For installation on CentOS/Rocky 8/9.x and Ubuntu 22.04 systems, see the Appendices of this document for system-specific instructions.
(The most recent package version number at this writing, for this OS, is shown):

• gcc → 4.8.5	
• gcc-g++ → 4.8.5	
• gcc-gfortran → 4.8.5	
• cmake**3** → 3.17.5	
• boost-devel → 1.53.0	
• openmpi**3**-devel → 3.1.3	*optional, for building multi-threaded version*
• gtest-devel → 1.6.0	
• xerces-c-devel → 3.1.1	
• hdf5-devel → 1.8.12	
• qt-devel → 4.8.7	*optional, for building GUI application*
• qwt-devel → 6.1.1	*optional, for building GUI application*

Before the build script is invoked on a CentOS 7.x system, the user's PATH environment variable may need to be updated to include paths to the necessary utilities, depending on the options chosen:

- if multi-threaded applications are to be built:	'/usr/lib64/openmpi3/bin'	(for 'mpirun')
- if GUI application to be built:	'/usr/lib64/qt4/bin'	(for 'qmake' utility)

Windows

The Windows executables and libraries are precompiled in the distribution, so the need for a build under Windows is rare; the directions are included here for completeness. Please note that IRENE does NOT support 32-bit Windows; all portions, including dependencies, must be built in 64-bit mode. It is recommended these dependencies and utilities be installed in the "*C:/Program Files*" location.

- HDF5 libraries version 1.10.2 or later
- Boost template library version version 1.53 or later
- Xerces-C++ library version 3.1.1 or later
- Intel MPI Library version 2022.1 or later *optional, for building multi-threaded version*
    - Obtain from https://software.intel.com/en-us/intel-mpi-library
    - *when VPN connection is active, requires env var setting:* `FI_TCP_IFACE = lo`
- Qt libraries version 6.6.0 or later *optional, for building GUI application*
    - *Includes the MinGW C++ compiler and associated pre-built Qt libraries*
- Qwt libraries version 6.2.0 or later *optional, for building GUI application*
    - The Qwt library has a strong dependence on the Qt library and its 'qmake' utility.
    - Some adjustment of the 'Irene/source/irene_gui/IreneGui.pro' file may be needed.
- CMake version 3.14 or later
- Microsoft Visual Studio 2022 (includes C++ compiler) *later versions may be used*

Required Build Tools

- CMake version 3.14 or later
- Python version 3.x scripting language *(required for automated build, some IRENE utilities)*

Build Procedure

A majority of the IRENE software build is performed using the CMake utility, and relies on its 'find_package' feature to locate those specific third-party libraries installed on the system.

The optional build of the GUI application includes the use of the 'qmake' utility, and the QT and Qwt library path information defined in the 'Irene/source/irene_gui/IreneGui.pro' file (preconfigured for CentOS 7.x). Other versions of this file for CentOS/Rocky 8/9.x and Ubuntu 22.04 are available; see instructions in the Appendix. For Windows builds, verify and update the specified paths as appropriate.

Pre-build Checklist

- Install all necessary utilities and third-party libraries
- Make PATH environment variable adjustments, if required
- Verify and update the GUI configuration file, if needed

Automated build

In a terminal window, change to the 'Irene/source' directory, then invoke the python (v3.x) build script, specifying the OS and any needed optional arguments:

```
  python buildIrene.py [ --nompi ] [ --nogui ] [--cmake=<alt>]
```

The operating system of the install is automatically determined. The debug mode may be specified using `'--mode=debug'`, but is strongly discouraged *(increases execution time by a factor of at least 50)*. By default, both the single- and multi-threaded versions of the programs are built, unless the optional argument '`--nompi`' is specified. The GUI application is also built by default; to skip this, specify the optional argument '`--nogui`'. The *cmake* is required to be at least version 3.14 (check via the command '`cmake --version`'); an alternate version of the utility by be specified with the optional argument.
→ For building on a CentOS 7.x system, add '`--cmake=cmake3`'.

The full build process will take several minutes, and will depend on options specified.

In the unlikely event that the CMake utility is unable to 'find' these third-party libraries, then the top-level 'CMakeLists.txt' files in the 'Irene/source/SpWx_Irene' and/or 'Irene/source/irene_cpp' directories will need to be edited. Prior to the relevant 'find_package' call in the script, the 'CMAKE_PREFIX_PATH' variable will need to be set to the appropriate library path, to enable the library to be properly 'found'.

When the build process is successful, all generated files are installed in the 'Irene/<platform>' directory and its subdirectories. Proceed to page 9 – *Testing of the Build*.

Manual Build for Linux

Because python is not essential for IRENE (AE9/AP9/SPM) and is not automatically installed on all operating systems, the following instructions are provided for building the model and applications manually. Note that the *cmake* utility is required to be version 3.14 or greater – substitute the appropriate alternate *cmake* name as needed (ie 'cmake3' on CentOS 7.x).

1. In a terminal window, change to the 'Irene/source' directory.
2. Create a directory 'build_linux' here, and change to that directory.
3. Create a directory 'Irene_SpWx' here, and change to that directory.
4. Enter the command: `cmake -DBUILD_IRENE_SPWX=ON ../../SpWx_Irene`
   For debug mode, also include the `-DCMAKE_DEBUG=Y` option.
   Warning: 'debug' mode increases program execution time by a factor of at least 50.
5. If no errors were reported during the cmake execution, enter the command: `make`
6. Some *warnings* may be seen during the compilation step. But if no errors were reported, then enter the command: `make install`

7. If no errors were reported during the installation step, navigate back to the ‘Irene/source/build_linux’ directory, which was created in step 2.

8. Create a directory ‘irene_cpp’ here, and change to that directory.

9. Enter the command: `cmake ../../irene_cpp`
   By default, both the single- and multi-threaded versions of the IRENE applications are built.
   To build *only* the single-threaded version, add the optional argument ‘`-DUSE_MPI=OFF`’.
   For debug mode, also include the `-DCMAKE_BUILD_TYPE=Debug` option.
   Warning: ‘debug’ mode increases program execution time by a factor of at least 50.

10. If no errors were reported during the cmake execution, enter the command: `make`

11. Some *warnings* may be seen during the compilation step. But if no errors were reported, then enter the command: `make install`

12. If no GUI application is desired, skip to step 18

13. If no errors were reported during the installation step, navigate back to the ‘Irene/source/build_linux’ directory, which was created in step 2.

14. Create a directory ‘irene_gui’ here, and change to that directory.

15. Enter the command: `cmake ../../irene_gui`
    If the only the single-threaded applications were built in step 9, similarly add the optional argument ‘`-DUSE_MPI=OFF`’, to disable the MPI-related controls within the GUI.

16. If no errors were reported during the cmake execution, enter the command: `make`

17. Some *warnings* may be seen during the compilation step. But if no errors were reported, then enter the command: `make install`

18. The build and install process is now complete. Proceed to page 9 – *Testing of the Build.*

Manual Build for Windows

Because python is not essential for IRENE (AE9/AP9/SPM) and is not automatically installed on all operating systems, the following instructions are provided for building the 64-bit Windows release-mode binaries model and applications manually. Note that IRENE is *not supported* for 32-bit Windows. Most cmake commands also require the option: ‘ `-G"Visual Studio 15 2017 Win64"` ’ (including double quotes). It is assumed that all third-party dependencies (listed on page 4) are already installed.

1. From a command prompt, navigate to the ‘Irene/source’ directory.

2. Create a directory ‘win64_release’, then navigate to that directory.

3. Create a directory ‘SpWx_Irene_win64_release’, then navigate to that directory.

4. Enter the command: `cmake -DBUILD_IRENE_SPWX=ON ../../SpWx_Irene`
   For debug mode, also include the `-DCMAKE_DEBUG=Y` option.
   Warning: ‘debug’ mode increases program execution time by a factor of at least 50.

5. If no errors were reported during the cmake execution, do the following:
   from Windows Explorer, double-click the newly-generated solution file ‘*SpWx.sln*’ in the ‘Irene/source/ SpWx_Irene_win64_release’ directory. From within Visual Studio, select the project “ALL_BUILD” in the project explorer pane. Verify that the build type is shown as ‘*Release*’ mode in the toolbar along the top. Select “Build ALL_BUILD” from the Build menu. Leave Visual Studio open.

6. Some *warnings* may be seen during the compilation step. But if no errors were reported, do the following:
   in Visual Studio right-click the ‘INSTALL’ project in the Solution Explorer and select ‘build’. After the operation is completed, close Visual Studio.

7. At the command prompt, navigate to the ‘Irene/source/win64_release’ directory

8. Create a directory ‘Irene_win64_release’ below it and then navigate to that directory.

9. Enter the command: `cmake ../../irene_cpp`
   By default, both the single- and multi-threaded versions of the programs are built.
   To build *only* the single-threaded version, add the optional argument ‘`-DUSE_MPI=OFF`’
   For debug mode, also include the `-DCMAKE_BUILD_TYPE=Debug` parameter.
   Warning: ‘debug’ mode increases program execution time by a factor of at least 50.

10. If no errors were reported during the cmake execution, do the following:
    from Windows Explorer, double-click the newly-generated solution file ‘*ae9ap9.sln*’ in ‘Irene/source/Irene_win64_release’. From Visual Studio, select project “ALL_BUILD”. Verify that the current build type is ‘*Release*’ mode in the toolbar along the top. Select “Build ALL_BUILD” from the Build menu. Leave Visual Studio open.

11. Some *warnings* may be seen during the compilation step. But if no errors were reported, do the following:
    in Visual Studio right-click the ‘INSTALL’ project in the Solution Explorer and select ‘build’. After the operation is completed, close Visual Studio.

12. If no GUI application is desired, skip to step 18.

13. If no errors were reported during the installation step, navigate back to the ‘Irene/source Irene_win64_release’ directory.

14. Create a directory ‘IreneGui_win64_release’ below and navigate to that directory.

15. The IreneGui application is built using the MinGW compiler (bundled with the QT installation, and its associated pre-built QT libraries). The PATH environment needs to include these directories (ie ‘*C:\Qt\6.6.0\mingw_64\bin*‘ and ‘*C:\Qt\Tools\mingw1120_64\bin*’).

16. Enter the command: `cmake -G “MinGW Makefiles” ../../irene_gui`
    If the only the single-threaded applications were built in step 9, similarly add the optional argument ‘`-DUSE_MPI=OFF`’, to disable the MPI-related controls within the GUI.

17. If no errors were reported during the cmake execution, enter the command: `mingw32-make`

18. Some *warnings* may be seen during the compilation step. But if no errors were reported, then enter the command: `mingw32-make install`.

19. The build and install process is now complete. Proceed to page 9 – *Testing of the Build*

Testing of the Build

At this point, the source distribution has been built and deployed to a location where it should run successfully using test input files. With the command prompt at the 'Irene/*<platform>*/bin' directory, verify the operational status of the IRENE software installation by entering this command:

```
  CmdLineIrene -i ../../samples/Ap9ShortInput.txt
```
← *add '* `-n 3`*' for testing multi-threaded mode*

The command-line utility should run successfully and produce output files in the 'Irene/samples' directory, matching the contents of files in the 'Irene/samples/expectedOutput' directory.

Similarly, the GUI application can also be invoked from this location with the command: `IreneGui` or, double-clicking its icon in the file manager window. Refer to the User's Guide for instructions and examples for using the GUI application.

If any errors are encountered during these tests, refer to the User's Guide, Appendix H for assistance. For SIGILL (signal 4) errors on Rocky 8/9.x systems, see bottom of Appendix A of this document.

Installation Customization
By default, the IRENE applications are executed from the platform's 'bin' directory. Alternatively, they can also be configured to be executed from another arbitrary directory; however, many of the model components depend on proper data file specifications; see the User's Guide, Appendix G for details.

Python-based Unit Tests
Also included in the distribution is a set of Python-based test scripts that can be used to verify the operation of each of the IRENE package model components. The use of Python 3.x is required, with the '*numpy*' and '*pytest*' modules installed. Prior to the execution of these scripts, add the top-level 'Irene' installation directory path to the *PYTHONPATH* environment variable; this enables the script to properly load the 'irene_defs.py' file, required for use of the Python API. To run the unit tests, navigate to the 'Irene/<*platform*>/bin' directory and enter the command:

```
  python -m pytest unitTest/
```

If errors are encountered, rerun the script, adding the '-vs' options for more detailed output:

```
  python -m pytest -vs unitTest/
```

Assistance with Build and/or Execution Problems
To report build and/or execution problems with the distribution, use the contact information found at the end of this document. Please specify distribution version number, platform (ie Windows vs Linux), and include any log file, input files and all error messages displayed. We will be unable to provide assistance without this information.

Other Platform Porting Successes
If successful in porting the software to a currently unsupported platform, please send a copy of the modified files, and a description of any other necessary steps/conditions, to the development team (ae9ap9@vdl.afrl.af.mil). This information will be greatly appreciated, and will be incorporated into future releases.

Appendix A: CentOS/Rocky 8/9.x Installation Notes

System Prerequisites

As a prerequisite to the installation of the required packages, the CentOS/Rocky 8/9.x system will need additional (distribution package) repositories. Use the following commands (as 'root' or via 'sudo').

- dnf install dnf-plugins-core
- dnf install epel-release
- *[Rocky 8.x only:]* dnf config-manager --set-enabled powertools *[Rocky 8.4 uses* 'PowerTools'*]*
- dnf update
- dnf repolist (resulting list should include: *epel, epel-modular, extras, [powertools])*
- dnf makecache --refresh

Required System Packages

Below is the list of system package names required to be installed on a CentOS/Rocky 8/9.x system for a successful build of the IRENE software. [The use of '*anaconda3*' packages is *<u>specifically NOT supported</u>.*]
(The most recent package version number at this time is shown, plus notable installed dependencies):

- gcc → 8.5.0-18
- gcc-c++ → 8.5.0-18 *Rocky 8.4 uses alternate name* 'gcc-g++'
- cmake → 3.20.2
- boost-devel → 1.66.0-13
- openmpi-devel → 4.1.1-3 *optional, for building multi-threaded version*
- gtest-devel → 1.8.0-5
- xerces-c-devel → 3.2.3-5
- hdf5-devel → 1.10.5-4
- hdf5-static → 1.10.5-4
- qwt-qt5-devel → 6.1.3 *optional, for building GUI application*
    - qt5-devel *(et al)* → 5.15.3 *optional, for building GUI application*

PATH Environment Variable Adjustments

Before the build script is invoked on a CentOS/Rocky 8/9.x system, the user's PATH environment variable may need to be updated to include paths to necessary utilities, depending on options chosen:

- if multi-threaded applications are to be built:	'/usr/lib64/openmpi/bin'	(for 'mpirun')
- if GUI application to be built:	'/usr/lib64/qt5/bin'	(for 'qmake' utility)

Special note for 'SIGILL' (signal 4, "illegal instruction") errors from multi-threaded runs

Due to an issue with the 'openmpi' dependencies, systems with older processors may fail with this error. To resolve this problem, install the "mpich-devel" package, then do a *full, clean* rebuild after revising the PATH environment variable to use '/usr/lib64/**mpich**/bin' instead of '/usr/lib64/openmpi/bin'.

Additionally, some installations using the 'mpich' MPI package may experience significant slowdowns; the multi-threaded mode of the test run shown at the top of page 9 typically completes in about 30 seconds or less. If it takes significantly more time to complete (apparently caused by repeated probes about hardware properties), some more actions are needed to eliminate these delays:

- install additional package:
    - `sudo dnf install hwloc-gui`
- generate a file containing hardware attributes for this particular system
    - `lstopo --of xml hwloc.xml`
    - place the resulting '*hwloc.xml*' file in an appropriate location
- set an environment variable to point to this file
    - `setenv HWLOC_XMLFILE /full/path/to/hwloc.xml`
    - `export HWLOC_XMLFILE=/full/path/to/hwloc.xml`

Following the generation of this file and the associated environment variable, the IRENE multi-threaded model runs should complete in their typical timeframes.

Appendix B: Ubuntu 22.04 Installation Notes

Required System Packages

Below is the list of system packages required to be installed on a Ubuntu (v22.04 LTS) system for a successful build of the IRENE software. [The use of '*anaconda3*' packages is *specifically NOT supported*.]
(The most recent package version number at this time is shown, plus notable installed dependencies):

- build-essential → (n/a)
  - make → 4.3-4
  - gcc → 11.3.0
  - g++ → 11.3.0
- cmake → 3.22.1
- libboost-all-dev → 1.7.4
  - libopenmpi-dev → 4.1.2
- googletest → 1.11.0
- libxerces-c-dev → 3.2.3
- libhdf5-dev → 1.10.7
- qtbase5-dev → 5.15.3 *optional, for building GUI application*
- qt5-qmake → 5.15.3 *optional, for building GUI application*
- libqwt-qt5-dev → 6.1.4 *optional, for building GUI application*
- python3-pip → 22.0.2
  - this is needed for installing python modules
  - modules to install: *pytest, xlsxwriter, numpy, matplotlib*, etc.

PATH Environment Variable Adjustments

No adjustments needed for Ubuntu.

Contact Information

Please send any questions, comments and/or bug reports to: ae9ap9@vdl.afrl.af.mil

The IRENE model package and related information can be obtained from AFRL's Virtual Distributed Laboratory (VDL) website: https://www.vdl.afrl.af.mil/programs/ae9ap9