SALOME Modules Porting Project for MacOS

Packaging and installation

12/01/2025

Customer : Polytechnique Montreal

Document reference : SIMVIA-POLY_MONTREAL_001

Technical contact at Simvia : basile.marchand@simvia.tech

Recipient(s) : Alain Hebert (alain.hebert@polymtl.ca)

Description :

This document aims to describe the installation and usage of the **Salome** platform on **macOS**, following the porting work that has been carried out. It provides an overview of the completed tasks, the technical choices made, and the procedures required to install and operate the platform in a macOS environment.

Document History

Version	*Date*	*Author(s)*	*Description*	*Reviewer*
1	12/01/2026	Basile Marchand	Initial version	
2	02/03/2026	Basile Marchand	Fix some installation issues	
3	06/03/2026	Basile Marchand	Add glow usage instructions	

Table des matières

Introduction

Installation and usage

System level dependencies

At the client's request, and in order to minimize the amount of software dependencies that must be managed manually, a large part of the prerequisites required for compiling and running **Salome** on **macOS** has been installed using **Homebrew**.

Throughout this document, it is assumed that **Homebrew** is already installed on the target system. If this is not the case, please refer to the official Homebrew documentation for installation instructions.

The following dependencies must be installed using **Homebrew**:

- **boost**
- **cppunit**
- **doxygen**
- **eigen@3**
- **hdf5**
- **libpng**
- **numpy**
- **omniorb**
- **pkgconf**
- **psutils**
- **pyqt-builder**
- **python-packaging**
- **python-setuptools**
- **qt@5**
- **qwt-qt5**
- **swig**
- **tbb**
- **sphinx**

Finally, in order to install a missing **Python** dependency, the following command must be executed after completing the **Homebrew** installations:

- `pip3.14 install --break-system-packages psutil PyQt5`

Resolving Library Version Conflicts (Homebrew)

In some cases, **Homebrew may install more recent versions of certain dependencies** (such as omniORB or HDF5) than those used during the SALOME build process.

This version mismatch can lead to runtime errors, typically related to missing dynamic libraries or unresolved symbols at startup.

SALOME is built against specific versions of:

- **HDF5**
- **omniORB 4.3.3**

If Homebrew installs newer versions, the macOS dynamic loader (dyld) may fail to locate the expected library filenames.

A practical workaround consists in creating symbolic links that redirect the expected library names to the installed versions.

Example: HDF5

```
ln -s /opt/homebrew/opt/hdf5/lib/libhdf5.dylib \
   /opt/homebrew/opt/hdf5/lib/libhdf5.3.1.0.dylib
```

SALOME may expect the following omniORB dynamic libraries:

- libomniORB4.3.3.dylib
- libomniDynamic4.3.3.dylib
- libomnithread.3.3.dylib
- libCOS4.3.3.dylib
- libCOSDynamic4.3.3.dylib

If Homebrew provides newer versions (e.g., 4.3.x without the exact suffix), create symbolic links accordingly.

Example:

```
ln -s /opt/homebrew/opt/omniORB/lib/libomniORB4.dylib \
   /opt/homebrew/opt/omniORB/lib/libomniORB4.3.3.dylib
```

```
ln -s /opt/homebrew/opt/omniORB/lib/libomniDynamic4.dylib \
    /opt/homebrew/opt/omniORB/lib/libomniDynamic4.3.3.dylib

ln -s /opt/homebrew/opt/omniORB/lib/libomnithread.dylib \
    /opt/homebrew/opt/omniORB/lib/libomnithread.3.3.dylib

ln -s /opt/homebrew/opt/omniORB/lib/libCOS4.dylib \
    /opt/homebrew/opt/omniORB/lib/libCOS4.3.3.dylib

ln -s /opt/homebrew/opt/omniORB/lib/libCOSDynamic4.dylib \
    /opt/homebrew/opt/omniORB/lib/libCOSDynamic4.3.3.dylib
```

⚠️ **Important**

This workaround assumes ABI compatibility between the installed Homebrew version and the version used at build time.
If major version differences exist, rebuilding SALOME against the locally installed libraries is strongly recommended.

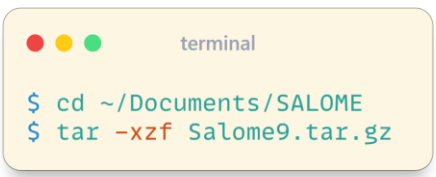
Salome distribution installation

Once the system-level dependencies have been installed, the installation of **Salome** itself can be performed.

Download the **Salome9.tar.gz** archive and extract it into a directory of your choice. **Important**: this directory must be kept permanently. The installation process does not copy Salome files into the *Applications* folder; the platform runs directly from this directory.

In the remainder of this document, it is assumed that the **Salome9.tar.gz** archive has been placed in the Documents/SALOME directory.

To extract the archive, run the following commands:



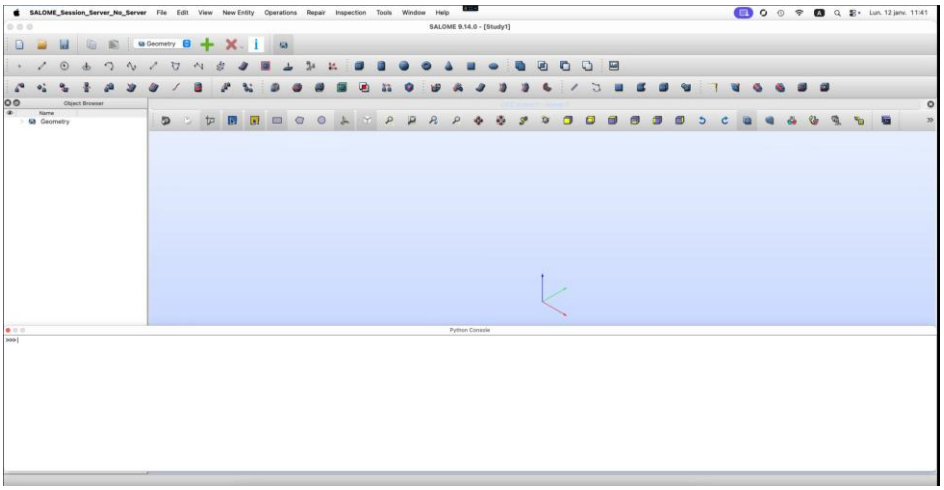
A final installation step is then required. This step is necessary due to the way **macOS** handles dynamic libraries. It only needs to be performed **once**. Run the following commands to complete the installation:

```
terminal

$ cd ~/Documents/SALOME/Salome9
$ chmod +x ./utils/install_rpath.sh
$ ./utils/install_rpath.sh
```

Salome distribution usage

Once the installation is complete, **Salome** can be launched by executing the salome.sh script located at the root of the Salome9 directory.

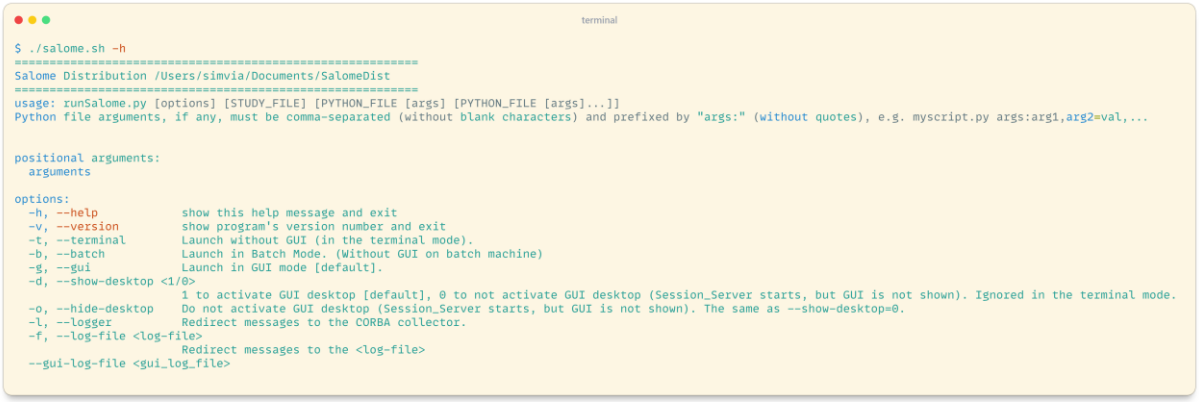
```
terminal

$ cd ~/Documents/SALOME/Salome9
$ ./salome.sh
```



For convenience, the ~/Documents/SALOME/Salome9 directory can be added to the PATH environment variable. To do so, add the following line at the end of your ~/.zshrc file:

```
terminal

export PATH=~/Documents/SALOME/Salome9:$PATH
```

The salome.sh script is a simple **wrapper** around the standard salome command commonly used on Linux systems. All options and arguments available on Linux are also supported in this macOS setup.



Managing macOS Security Restrictions

Depending on the macOS version and security configuration, launching the SALOME executable (which is not code-signed) may trigger security warnings or result in certain components being automatically quarantined by the operating system.

To ensure proper execution, the following steps may be required:

1. **Temporarily Allow Applications from Any Developer**
   a. Open a terminal and execute:

   sudo spctl --master-disable

   This command enables the **"Anywhere"** option in macOS Gatekeeper settings.

   Then:

   - Open **System Settings**
   - Navigate to **Privacy & Security**
   - Under **Security**, select **Allow applications downloaded from: Anywhere**

   ⚠️ This setting reduces the system security level.

2. **Remove the Quarantine Attribute from SALOME**

   macOS may automatically assign a quarantine attribute to unsigned applications and their internal libraries.

   To remove this attribute:

   1. Open a terminal.
   2. Navigate to the SALOME installation directory (e.g., Salome9).
   3. Run:

   xattr -dr com.apple.quarantine $(pwd)

This command recursively removes the quarantine flag from all files within the current directory.

After completing these steps, SALOME should launch without additional security prompts.

GLOW usage

In this section, we describe how to use the GLOW module with SALOME on macOS.

First, retrieve the GLOW module source code from the following repository:

https://github.com/newcleo-dev-team/glow

In the remainder of this document, we assume that the GLOW sources are located in the directory:

/Users/username/glow

Before launching SALOME, the Python environment must be configured so that Python can locate the GLOW module. This is done by extending the PYTHONPATH environment variable as follows:

export PYTHONPATH=/Users/username/glow:$PYTHONPATH

Once this environment variable has been set, SALOME can be started from the terminal using the standard launch command:

salome.sh

Once SALOME is running, a GLOW script can be executed directly from the graphical interface. To do so, open the **File** menu and select **“Load Script”**, then specify the Python script to execute. For instance, you may load the example script colorset.py.

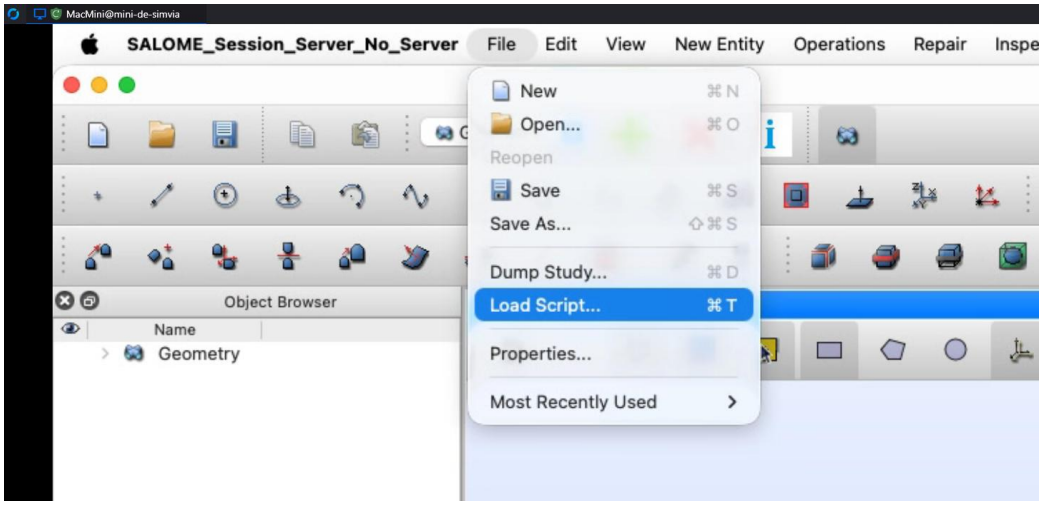


Table of Actions

N°	Deadline	Description	Who	Statut
1	12/01/2026	Delivery of a **macOS** version of **Salome** including a fully functional **GEOM** module.	Simvia	☒
2		Receipt, review, and validation of the delivered version.	Polytechnique Montreal	☒