1818 Library ◆ Suite 600 ◆ Reston, VA 20190-5602 ◆ (703) 326-2861  
<http://www.metsci.com>



Building Installers

V1

Oct 24, 2013

Thomas M. Kratzke

Sr. Analyst

Contents

[1. Introduction 1](#_Toc372698533)

[2. Windows Explorer 2](#_Toc372698534)

[2.1 Prepare Sample Directory:Clean out sample files. 2](#_Toc372698535)

[2.2 Clean out Maven Directories. 2](#_Toc372698536)

[2.3 Build an Archive Zip File for the C++ Projects. 2](#_Toc372698537)

[3. Work within Eclipse 3](#_Toc372698538)

[3.1 Update Maven jar files. 3](#_Toc372698539)

[3.2 Remove Etopo and Gshhs Data Jars from the Classpath of Both Sim64Git and Sim32Git. 3](#_Toc372698540)

[3.3 Set Version.Name. 3](#_Toc372698541)

[3.4 Eclipse part of SimLib01 preparation 3](#_Toc372698542)

[3.4.1 Build the Java Symbols for SimLib01. 3](#_Toc372698543)

[3.4.2 Build the Jar File and Folder of required Jars for SimLib01. 4](#_Toc372698544)

[3.4.3 Step 1: Create the directory of Utility jar files. 4](#_Toc372698545)

[3.4.4 Step 2: Over-write the SimLib01 jar file with source/classes jar file. 4](#_Toc372698546)

[3.5 Eclipse part of Sim64 and Sim32. 4](#_Toc372698547)

[3.5.1 Create folder of utility jar files. 4](#_Toc372698548)

[3.5.2 Write the “a” and “b” versions of the Main Jar File. 5](#_Toc372698549)

[3.6 Eclipse part of MathLib01 (64bit)\*\* 5](#_Toc372698550)

[3.7 Eclipse part of MathLib01 (32bit) 5](#_Toc372698551)

[4. Work within MSVC 6](#_Toc372698552)

[4.1 Build MathLib01 6](#_Toc372698553)

[4.2 Build SimLib01 6](#_Toc372698554)

[5. Work within Advanced Installer 7](#_Toc372698555)

[5.1 Build Sim64Installer 7](#_Toc372698556)

[5.1.1 Go to Files/Folders and select lib. 7](#_Toc372698557)

[5.1.2 Go to MiscArchive 7](#_Toc372698558)

[5.1.3 Global Settings for all Java products 7](#_Toc372698559)

[5.1.4 Go to Media 7](#_Toc372698560)

[5.1.5 Go back to MSVC 7](#_Toc372698561)

[5.2 Build Sim32Installer 7](#_Toc372698562)

[5.2.1 Go to Files/Folders and select lib. 7](#_Toc372698563)

[5.2.2 Global Settings for all Java products 7](#_Toc372698564)

[5.2.3 Go to Media 7](#_Toc372698565)

[5.2.4 Go back to MSVC 7](#_Toc372698566)

List of Algorithms

List of Code

**No table of figures entries found.**

List of Figures

**No table of figures entries found.**

List of Formulas

**No table of figures entries found.**

List of Snippets

**No table of figures entries found.**

List of Tables

**No table of figures entries found.**

Document Revision History

|  |  |
| --- | --- |
| Version Date | Comments |
| Oct 24, 2013 | For *Sim* V1.5 |
| Nov 20, 2013 | For *Sim* V1.5 |
|  |  |
|  |  |
|  |  |

# Introduction

Suppose the C++ and Java code are at a point where you wish to put together the installers. This document describes how you would then use Eclipse, Visual C++, and Advanced Installer to build the installers.

This document discusses how to build the 64bit installer. Sections marked with a \*\* must be repeated for building the 32bit installer. It is fairly straightforward to figure out how to do this. We recommend that you follow the directions, and then go through the \*\* ones for the 32bit installer.

There are several “chunks” of work. Roughly speaking these chunks are:

1. Work within Eclipse
2. Work within MSVC
3. Work within Advanced Installer, including building an installer and running it
4. Testing SimLib01 within MSVC
5. Testing installed version of Sim/Planner

# Windows Explorer

## Prepare Sample Directory:Clean out sample files.

Note that this does not need to be re-done for 32bit.

Start by using Windows Explorer. Go to D:\kratzke\Sarops\RunDirectory and clean up the RunDirectory/Cases directory. You can exclude cases in the directory RunDirectory/Cases by putting them in a folder whose name starts with “(IGNORE.” Such cases will not be included with the install.

While using Windows Explorer, double-check Sim.properties and SimCaseManager.properties in the RunDirectory/Data folder. Most of the time, you will want the following and only the following lines to not be commented out in Sim.properties:

SIM.use\_UI.OVERRIDE = true  
PLAN.use\_UI.OVERRIDE = true  
SIM.always\_run\_simulator.DEFAULT = true  
PLAN.useDetailedGetStatus.OVERRIDE = true

The file RunDirectory/Data/SimCaseManager.properties should have the following lines:

SimCaseManager.MaxNProcessorsToUse=95  
SimCaseManager.MaxNParticleFiles=2  
SimCaseManager.MaxNEngines=2  
SimCaseManager.MinNThreadsPerEngine=3

## Clean out Maven Directories.

Note that eclipse must *not* be running for this step. If it is, close it down. In this step, we rename a directory that eclipse will be using, and windows explorer will not allow us to do this if some program is using that directory.

Go to d:\JavaLibraries, and rename Repository to RepositoryX. When the Maven dependencies are rebuilt, you can delete RepositoryX.

## Build an Archive Zip File for the C++ Projects.

Put copies of the entire folders SimLib01 and MathLib01 into some convenient folder; call it convenientFolder. In the copies, delete any directory ipch, debug, release, or x64. Also, delete any sdf file. Then zip convenientFolder into convenientFolder.zip. We will include this in the miscellaneous archives part of Advanced Installer.

# Work within Eclipse

## Update Maven jar files.

Note that this does not need to be re-done for 32bit. Right-click on the project Sarops64Git, select Maven/Update Project, and then click the box “Force Updates of Snapshots/Releases” before clicking the box “OK.” Much time will go by as Maven retrieves the most recent versions of the 3rd party jar files.

## Remove Etopo and Gshhs Data Jars from the Classpath of Both Sim64Git and Sim32Git.

Note that this does not need to be re-done for 32bit. If we do not do this, then editing the dependencies directory within Advanced Installer, to exclude these files, is necessary. The latter step is already done, so this is just a “belt-and-suspenders” step.

## Set Version.Name.

This does not require re-doing for 32bit. This step is to mark the cases run with this installer with the version of the jar file.

Go to the file com.metsci.sarops.util.SimGlobalStrings.properties and set the “version.Name” line appropriately. In the rest of this document, we will assume that Version.Name is set to 1.5a-Oct24\_2013.

Occasionally in this document, we will set Version.Name to 1.5b-Oct24\_2013. We recommend always re-setting it to the “A” version as soon as the step requiring the “B” version is complete.

## Eclipse part of SimLib01 preparation

Note that this does not need to be re-done for 32bit. SimLib01 is a C++ library that turns around and and uses java classes. It is used by other C++ programs outside of SAROPS. It’s provided as a service for the rest of the SAROPS team.

### Build the Java Symbols for SimLib01.

This step is optional since these symbols do not need building. We build them only to refresh a reference file in the MSVC project. The symbols were copy/pasted from the reference file directly into the MSVC code.

We build the symbols necessary by using the Eclipse External Tool MakeJavaSymbols. It is configured as follows:

1. Location: ${Jdk64}bin\javap.exe.
2. Working Directory: ${workspace\_loc:/Sarops64Git/target/classes}.
3. Arguments: -version -public -s -classpath . com.metsci.sarops.planner.util.LadderPattern com.metsci.util.navigation.NavigationCalculator

javap does not have an option for specifying the output so we simply copy the results from the console and put it into a (reference only) text file within the MSVC project SimLib01.

### Build the Jar File and Folder of required Jars for SimLib01.

The purpose of this is to create a jar file with our source and a directory of all of the jar files that SimLib01 requires.

### Step 1: Create the directory of Utility jar files.

We use the export Runnable Jar tool. Right-click on the project SimLib01 and choose “Export,” then “Runnable Jar File.” Fill in the options as follows, making appropriate changes:

1. Click on “Copy Required libraries into a sub-folder next to the generated JAR.”
2. Launch Configuration: Select “LadderPattern – SimLib01.”
3. Export Destination: D:\kratzke\SAROPS\lib\SimLib01Exports\SimLib01-1.5-Oct24\_2013.jar
4. Do not save the ANT Script.

Then go to D:\kratzke\SAROPS\lib\SimLib01Exports and you should see a jar file and a directory, both named SimLib01-1.5-Oct24\_2013. The next thing we will do is overwrite the jar file.

Note that there is no “a” or “b” for SimLib01.

### Step 2: Over-write the SimLib01 jar file with source/classes jar file.

We use the pre-stored jardesc file from the project SimLib01:

1. Double-click SimLib01.jardesc from the project SimLib01.
2. Make sure that resources, class files, and source are all being exported.
3. Specify the Jar file name to overwrite the main jar file created in Step 1. (e.g. D:\kratzke\SAROPS\lib\SimLib01Exports\SimLib01-1.5-Oct24\_2013.jar)
4. Next, Next, Finish, Yes, Yes.

## Eclipse part of Sim64 and Sim32.

This part is very similar except that $Jdk64 is replaced by $Jdk32 and work will take place within the project Sarops32.

### Create folder of utility jar files.

Use the Runnable Jar tool as with SimLib01, with the following modifications:

1. Launch Configuration: MathX32 – Sarops64Git (or MathX32 – Sarops32Git).
2. Export Destination: D:\kratzke\SAROPS\lib\SimExports\Sim64-1.5-Oct24\_2013.jar (or D:\kratzke\SAROPS\lib\SimExports\Sim32-1.5-Oct24\_2013.jar).
3. Don’t save the ANT configuration.

Then go to the directory and delete the isolated jar file, leaving a newly created folder of jar files.

### Write the “a” and “b” versions of the Main Jar File.

This does not need re-doing for 32bit. Change the Version.Name to 1.5b-Oct24\_2013. Double-click Sim.jardesc and make sure the name of the jar file is correct (including the “b” part of the name).

Change the Version.Name back to 1.5a-Oct24\_2013, and double-click Sim.jardesc again, changing the name of the jar file back to include the “a.”

## Eclipse part of MathLib01 (64bit)\*\*

You will need to repeat this section for the 32bit product. MathLib01 is a “helper” C++ library that needs to be built to satisfy the “native” references within the two java classes com.metsci.util.navigation.GreatCircleArc and com.metsci.util.MathX.

The Eclipse part of this project uses an external tool to generate the header files for the C++ code. The external tool is configured as follows:

1. Location: ${Jdk64}bin\javah.exe (where Jdk64 is set to the directory of the 64bit Jdk, including the final slash).
2. Working Directory: ${workspace\_loc:/Sarops64Git/target/classes} (or wherever the “.class files” are stored).
3. Arguments: -d "C:\Users\kratzke\Documents\CVSRoot\kratzke\trunk\MathLib01\MathLib01\GeneratedHdrs64" -force -verbose -classpath . com.metsci.util.MathX com.metsci.util.navigation.GreatCircleArc

The external tool was saved as “GenerateHdrs64.”

## Eclipse part of MathLib01 (32bit)

This is the same as the previous section except:

1. Use ${Jdk32} instead of ${Jdk64}
2. The Working directory is “…/Sarops32Git…” instead of “…/Sarops64Git….”
3. The Arguments should have “…GeneratedHdrs32” instead of “…GeneratedHdrs64.”

# Work within MSVC

## Build MathLib01

Open the solution MathLib01. Change the View to PropertyManager and double-check that Jdk32 and Jdk64 are correctly set. Switch back to the Solution view.

Go to MathLib01 properties and make sure that:

1. for All Configurations/Win32, the include directories read:  
   $(Jdk32)include;$(Jdk32)include\win32;$(ProjectDir);$(ProjectDir)GeneratedHdrs32;$(ProjectDir)MyHdrs;%(AdditionalIncludeDirectories).
2. for All Configurations/x64, the include directories read:  
   $(Jdk64)include;$(Jdk64)include\win32;$(ProjectDir);$(ProjectDir)GeneratedHdrs64;$(ProjectDir)MyHdrs;%(AdditionalIncludeDirectories)

Do a Batch Build and build all 4 of them. Check that all 4 succeed.

## Build SimLib01

Open the solution SimLib01. Change the View to PropertyManager and double-check that Jdk32 and Jdk64 are correctly set, and that Jre32 and Jre64 are set to the deployed versions. Switch back to the Solution view.

Go to SimLib01 properties and make sure that:

1. For All Configurations/Win32, the include directories have the Jdk32 includes
2. For All Configurations/x64, the include directories have the Jdk64 includes

Do a Batch Build and build all 8 of them. Check that all 8 succeed.

# Work within Advanced Installer

## Build Sim64Installer

### Go to Files/Folders and select lib.

Right-click on the Dependencies folder and synchronize to the 64bit SimExport folder you built from within Eclipse.

Add the “a” jar you built in Eclipse. Add the “b” jar, but rename it by changing its extension to zip.

### Go to MiscArchive

Delete the zip file that is the archive for the C++ projects and include the one referred to above as “convenientFolder.zip.”

### Global Settings for all Java products

Above the “Products” node in the middle of the screen, there’s a “Settings” tab. Be sure that, when the “Jre Bundle” is edited, that the Jre is set correctly for 64 and “none” is selected for 32.

### Go to Media

Change the build to Oct24\_2013, and change the MSI and EXE names.

### Go back to MSVC

Run SimLib01’s solution’s debugging run (ConsoleApp). It should run and execute the SimLib01 from the deployed directory.

## Build Sim32Installer

### Go to Files/Folders and select lib.

Right-click on the Dependencies folder and synchronize to the 32bit SimExport folder you built from within Eclipse.

Add the A jar you built in Eclipse. Add the B jar, but rename it by changing its extension to zip.

### Global Settings for all Java products

Above the “Products” node in the middle of the screen, there’s a “Settings” tab. Be sure that, when the “Jre Bundle” is edited, that the Jre is set correctly for 32 and “none” is selected for 64.

### Go to Media

Change the build to Oct24\_2013, and change the MSI and EXE names.

### Go back to MSVC

Run SimLib01’s solution’s debugging run (ConsoleApp). It should run and execute the SimLib01 from the deployed directory.