Compiling *Silbido*

If you downloaded *silbido* from the distribution web site, chances are you already have compiled classes. However, if you download source code only, e.g. from a repository, there are several parts of *silbido* that must be compiled. *Silbido* uses a combination of Matlab, Java, and C++ code.

Matlab compilation

There are several C/C++ programs that interface with Matlab. In directory src/matlab/lib/audio/stat there are two moving average functions stMA.c and stMARestricted.c. In src/matlab/lib there is tineyxml2_wrap.cpp. All three of these files need to be compiled using mex. This requires a C/C++ compiler supported by Matlab. Details on supported compilers is available at Mathworks. If you are not running the most recent Matlab, check your version documentation for the list. Recent versions of Matlab will let you install a supported compiler from Matlab.

Each of these files should be compiled by changing directory (cd) to the directory containing the file and typing "mex" followed by the filename, e.g.:

```
>> mex tinyxml2_wrap.cpp
```

The first time you run mex, you will need to use mex -setup to let Matlab learn where your installed compiler is located.

Building Java classes from source

If you downloaded *silbido* from the distribution web site, chances are you already have compiled classes for the Java code. However, if you see the following message when initializing *silbido*:

```
Could not load java classes.
Please ensure that they have been compiled into directory
[SilbidoRootDirectory]/silbido/src/java/bin
with the appropriate Java architecture.  Use "ver java" to find the
version of Java that Matlab is using.
```

then either there is a version mismatch between your Java classes or they have not been compiled at all. Check the silbido/src/java/bin directory to see if it exists and is populated with .class and .jar files. If it is not, these need to be built, and can be done so with a Java compiler and Ant.

Matlab has used different versions of Java over the years. Starting with R2017b, they use Java 1.8. Java 1.7 (frequently just called Java 7) was used starting R2013b. Prior to that, Java 1.6 (or just 6) was the Java deployed with Matlab for many years. Check the version of Java Matlab expects:

```
>> ver java
-----------------------------------------------------------------------
MATLAB Version: 8.2.0.701 (R2013b)
MATLAB License Number: 171045
```

```
Operating System: Microsoft Windows 7 Version 6.1 (Build 7601: Service
Pack 1)
Java Version: Java 1.7.0_11-b21 with Oracle Corporation Java
HotSpot(TM) 64-Bit Server VM mixed mode
-------------------------------------------------------------------------
```

In this example, we are using Java 7. In general, we will need to set a compile "target" for the version that we are using. Java 6 binary classes might work for Java 7 (untested), but the other way around will fail without giving a meaningful error message.

There are a number of ways to compile the Java source. In this example, we will require:

- Ant – a compilation manager developed by The Apache Organization that can be downloaded at: http://ant.apache.org/
- A Java development kit, available from Oracle: http://www.oracle.com/technetwork/java/index.html (download the standard edition, Java SE).

Apache does not have an installer, it is simply unarchived into a directory. The Java development kit will have a normal installer (for Windows).

Once you have set everything up, open a command window and use "cd" to change to the directory where *silbido* is installed (the paths used here will depend on where you installed things):

cd C:\Users\YourAccount\Documents\matlab\silbido

Set an environment variable with the directory where Java has been installed. In this example, it was installed to c:\apps\develop\java:

```
set JAVA_HOME=C:\apps\Develop\java
```

Then run ant:

```
[PathToAntDir]\bin\ant compile
```