Building *silbido* from source

The *silbido* tonal detector is written in a combination of Matlab, Java, and C++. If you downloaded a distribution package from our web site, you don't need to be reading this unless you are running an operating system for which we have not built the C++ code.

Build requirements

To build *silbido*, you will need the following:

- A C++ compiler supported by your current version of Matlab. To determine which compilers your version of Matlab supports see the documentation for the mex command. There is a section in the documentation about supported and compatabile compilers. Compiler installation on most linux platforms can be accomplished with your package manager used by your linux distribution (e.g. yum, dpkg, etc.). Microsoft provides a free version of their Visual Studio C++ compiler.
- A Java development kit (JDK) that can be downloaded from Oracle. Note that if you attempt to use a JDK newer than the Java compiler in Matlab without setting compatibility options, you may encounter problems. You can determine Matlab's Java version with the command "ver –java".
- Apache's ant build tool. Follow the installation instructions to ensure that paths and environment variables are all set.

Building Java products

Start a console window and change directory to the root of the *silbido* distribution, which we will call $root through the rest of this section. To compile the Java code, type ant followed by the target name:

- ant compile – Will compile all Java files to $root/build. The 3rd party library files may be found in the $root/build/jar subdirectory and the silbido tonals package in directory $root/build/java/tonals. Note that silbido_init will look for these directories first.
- ant buildJar – Builds a Jar file with classes specific to *silbido* In the $root/build/jar file. This jar contains anything that is in the $root/build/java directories and will invoke the compile target if it is out of date.

The following build targets are for creating packages to be redistributed. The first target simply copies everything into a clean directory without any of the source control whereas the second target build compressed archives. Before executing the following two build targets, it is recommended that the C++ targets be compiled so that they are included in any distributions that are created.

- ant package – Invokes compile, buildjar, and then creates a distributable in $root/build/package.
- ant dist – Generates compressed (zip and gzip) products suitable for distribution. These are placed in $root/dist.

Buiding C++ products

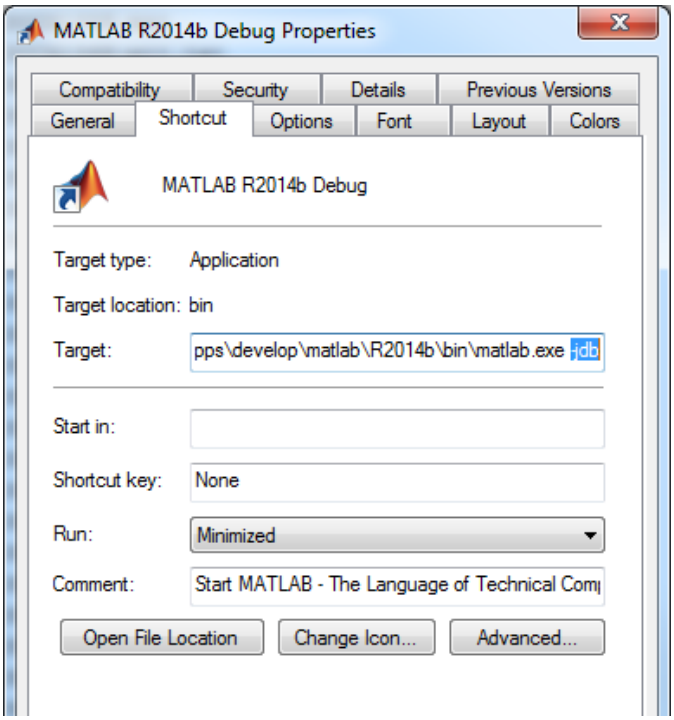
If you have not configured your C++ compiler for use with Matlab, type “mex –setup” at the Matlab prompt. This need only be done once.

Relative to the root of the *silbido* distribution, set Matlab’s current working directory to src/matlab/lib.

Type “mex tinyxml2_wrap.cpp”. This will compile a Matlab executable whose extension depends on the operating system being used. For example, 64 bit Windows currently will produce “tinyxml2_wrap.mexw64”. For the curious, Matlab’s mexext function will indicate the extension that will be used.

Notes on debugging *silbido* Java code

It is very helpful to be able to run a Java debugger when developing new functionality for *silbido*. We recommend using eclipse, but jdb or any debugger will do. When starting Matlab, provide the –jdb option to the startup. When Matlab is started from a graphical launcher (e.g. desktop menu), you may need to create a modified copy of the launcher. For example. on Windows find the Matlab startup icon and create a copy. Then edit the properties and add –jdb:



When Matlab is opened with this launcher, you will see a message similar to the following one:

```
JVM is being started with debugging enabled.
Use "jdb -connect com.sun.jdi.SocketAttach:port=4444" to attach debugger.
```

To specify a different port, add the port number after the –jdb. To debug using eclipse simply import the silbido project from the Hg repository. Then set up a new remote Java debugger with the specified

port. We use eclipse wth the Mercurial Eclipse project and TortoiseHg for development, but jdb is fine too.