

The *silbido* file format

Silbido is the graph-based whistle detector described in Roch et al. (2011) and has a binary file format. The file format is described below along with instructions on how to use provided software to store and read tonals using either Matlab or Java. Tonals are represented as lists of time x frequency nodes where time is in seconds since the start of a file or stream and frequency is in Hz.

The most recent release of the file format supports the optional storage of signal to noise ratio and phase for each node of a tonal and permits a score and/or confidence value to be saved with each tonal.

Matlab

Prior to using Matlab, it is necessary to ensure that Matlab can find the Java classes and Matlab files that implement reading and writing. Assume that variable *filefmt* contains the top level directory of the package that was downloaded and contains the Matlab .m files.

Upon starting Matlab, execute the following (this must be done each time Matlab is started, you can also add these to your startup.m file):

```
addpath(filefmt)
javaaddpath(filefmt);
```

The *dtTonalsLoad()* and *dtTonalsSave()* files are used to read and write sets of tonals. The *Manipulating Tonals* help document demonstrates how to create and use lists of tonals. A concrete example can be seen in *tonal_test.m*, a Matlab function that tests the functionality of the functions and also shows how to use the load and save functions.

It is also possible to call the Java functions directly which allows one to write detections incrementally. Examples of how to do this can be seen in the file *dclde2013_demo.m*.

Java

The two classes used to read and write tonal streams are *TonalBinaryInputStream* and *TonalBinaryOutputStream*.

The *TonalBinaryOutputStream* expects a constructor containing the filename, a version number associated with the detections (put 0 if you don't care), a comment, and a bit mask which indicates what should be stored. Bit mask values are constructed from bitwise or of the following flags associated with the class *TonalHeader*: TIME, FREQ, SNR, PHASE, SCORE, and CONFIDENCE. Together, they describe what will be written for each tonal.

Several write methods are available for *TonalBinaryOutputStream* objects, they expect either *Silbido* tonal objects or type double arrays of time and frequency. Methods ending in *_c*, *_s*, *_cs* expect

confidence and/or scores in addition to the tonal contour information. The close() method completes the stream.

The TonalBinaryInputStream simply expects a filename and supports the following methods:

- getTonals() – Returns a linked list of tonals.
- getConfidences() – Returns a double array of confidence scores (if they were saved)
- getScores() – Returns a double array of scores (if they were saved)
- getHeader() – Returns a TonalHeader instance which has other methods such as getComment() and getUserVersion().

One can read the Matlab code for examples on how to use these classes as the syntax for using Java classes in Matlab is nearly identical to coding in Java.

The file format

The following shows a hexadecimal dump of a Silbido save of two short contours with two time-frequency nodes each and scores associated with them:

```
87654321: 0011 2233 4455 6677 8899 aabb ccdd eeff
00000000: 7369 6c62 6964 6f21 0002 0013 0001 0000 silbido!.....
00000010: 0039 0025 6d79 2032 6e64 2067 656e 6572 .9.%my 2nd gener
00000020: 6174 696f 6e20 6465 7465 6374 6f72 2f63 ation detector/c
00000030: 6c61 7373 6966 6965 7240 5200 0000 0000 lassifier@R.....
00000040: 0000 0000 0240 3133 3333 3333 3340 4400 .....@1333333@D.
00000050: 0000 0000 0040 3300 0000 0000 0040 4e00 .....@3.....@N.
00000060: 0000 0000 0040 4880 0000 0000 0000 0000 .....@H.....
00000070: 0240 3900 0000 0000 0040 4900 0000 0000 .@9.....@I.....
00000080: 0040 3c00 0000 0000 0040 4e00 0000 0000 .@<.....@N.....
00000090: 00
```

This is a hexadecimal (base 16) dump, the leftmost number shows the number of bytes into the file in base 16, and every pair of numbers represents a byte of the file. For example, the byte stored at position 18_{16} (24_{10}), can be found in the second row of data (00000010) in the 88 column and contains the value $6e_{16}$. When numbers span more than one byte (e.g. 2 and 4 byte integers and 8 byte doubles), values are saved from the most significant byte to the least (big-endian format).

- Bytes 0-7 contain the string “silbido!” in UTF-8/ASCII encoding
- Bytes 8-9 contain the Silbido file format version: 2
- Bytes a-b contain the bit mask: $13_{16} \rightarrow 10011_2$ which corresponds to storing time, frequency, and score (see the constants in TonalHeader.java).
- Bytes c-d contain a version number provided by the user and associated with their detection algorithm: 1.
- Bytes e-11 contain the number of bytes in the header including the comment that follows the header size: $39_{16} \rightarrow 57_{10}$. So the header information occupies bytes 0_{16} to 38_{16} .
- The comment is present if the header size is longer than 12_{16} bytes. This is written in Java’s modified UTF8 coding format which uses two bytes to determine how many characters are in a

string (25₁₆) which is the length of “my 2nd generation detector/classifier” and then writes the characters themselves.

The tonals follow the header. Each tonal has the following format:

- If a score is to be saved, it is saved as an 8 byte double precision number.
- If a confidence is to be saved, it is saved as an 8 byte double precision number.
- A 4 byte integer indicates how many elements there are in the current tonal. For each of these elements, 8 byte double precision numbers are stored for each of the time x frequency node flags that are set in the bit mask in the following order: time, frequency, SNR, phase.

In the above example, the score of 72 is saved in bytes 39₁₆-40₁₆. Bytes 41₁₆-44₁₆ contain the value 2 indicating there are two data points associated with this tonal. For each of those, we read an 8 byte time and 8 byte frequency. A second tonal is also contained in this file.