autoBWF

Release 3.4.0

Feb 19, 2020

Contents

autoBWF is a Python/PyQt5 GUI for embedding internal metadata in WAVE audio files using the Broadcast Wave standard, FADGI BWFMetaEdit, and XMP.

Unlike the existing BWFMetaEdit GUI, autoBWF is extremely opinionated and will automatically generate metadata content based on file naming conventions, system metadata, and pre-configured repository defaults. In addition, it can copy metadata fields from a template file to avoid having to enter the same information multiple times for derivative files of the same physical instantiation.

Also included are auxilliary command line scripts which simplify the creation of derivative files and provide for export of embedded metadata to PBCore and CSV files.

CHAPTER 1

Contents

1.1 Installation

1.1.1 Prerequisites

Python 3

This code requires Python 3.6 or later. If you are running on Linux or Mac OS, then it is almost certainly already installed. If you are running on Widows, you can install Python using instructions on any number of web sites (such as this one).

You may wish to configure a virtual environment within which to install and run `autoBWF`.

bwfmetaedit

If needed, download and run the installer for the BWFMetaEdit CLI (Command Line Interface) appropriate to your operating system from mediaarea.net. Note that having the BWFMetaEdit GUI installed is not sufficient.

`autoBWF` has been tested with `bwfmetaedit` v1.3.3 and v1.3.8. Note that earlier versions of `bwfmetaedit` have a bug that introduces spurious characters at the end of the `CodingHistory` element. This bug has been confirmed to exist in v1.3.1.1, and it may affect other versions prior to v1.3.3.

Optional software

In order to run `autolame` and `autosplice`, you will also need to install lame (for `autolame`) and SoX v14.4.2 (for `autosplice`).

Note that some LINUX package repositories (e.g. Ubuntu 16.04) have an earlier version of `SoX` that seems to have problems with time specifications, so you may need to install from source. If you are doing so on Ubuntu, you may need to run `sudo ldconfig` after installation if you get a `sox: error while loading shared libraries: libsox.so.3: cannot open shared object file: No such file or directory` error.

1.1.2 Installing autoBWF

The latest release of `autoBWF` is available on PyPI and can be installed using

```
pip3 install autoBWF
```

The "bleeding edge" version is in the master branch on github, and can be installed by cloning the repository and installing the local code with `pip3`, or by running

```
pip3 install git+git://github.com/Ukrainian-History/autoBWF.git#egg=autoBWF
```

The master branch *should* contain functional code — development work that is likely to result in a broken state is done on feature branches.

1.2 Usage

1.2.1 Opening target file and editing metadata

Simply type

```
autoBWF
```

on the command line, and use the GUI "Open" menu item to load your WAVE file.

Alternatively, you can avoid the "Open" dialog by specifying the target file on the command line:

```
autoBWF <target_filename>
```

autoBWF will prepopulate the Description, Originator, OriginationDate, OriginationTime, and OriginatorRef GUI elements with reasonable guesses as described below.

If <*target_filename*> already contains embedded metadata, then those values will appear in the GUI as grey italic text. If the user edits those fields, the text color will change to red as a warning that the original values will be overwritten in the target file. **This cannot be undone!** You can use the drop-down menu to the right of or below each text element to switch between the original and edited versions prior to saving.

The CodingHistory text is generated automatically based on the selections made in the drop-downs to the right of the text box and the values in the *configuration* file. Similarly, the Copyright text is replaced with the boilerplate corresponding to the dropdown menu selection.

Note: You can edit the Copyright and Coding History texts manually prior to saving, but be aware that using the drop-down menus will destroy any manual edits that you have made.

1.2.2 Template files

autoBWF supports the transfer of metadata embedded in one file (the "template" file) to the target file. This can be done either by clicking on the "Load Template" button, or by using the optional `-t` command line argument:

```
autoBWF <target_filename> -t <template_filename>
```

This will prepopulate the contents of the Title, Technician, Source, Copyright, Coding History, Archival Location, and all XMP data fields with the corresponding metadata contained in *template_filename*. These can always be edited before saving the metadata to the target file. If the target file has existing metadata that conflicts with the template, then the template value text will be shown in yellow italics. The drop-down menus next to each field can be used to switch between the existing, template, and edited texts.

1.2.3 Saving metadata

When you have entered or modified the data and reviewed it for accuracy, you can save those values to the target file by clicking on "Save metadata" in the top menu bar. Again, once you click "Save metadata", all changed metadata fields will be overwritten without any possibility of recovery.

Note: It is strongly recommended that you work with test files and confirm (using BWFMetaEdit and/or a metadata viewer like exiftool) that autoBWF is behaving the way that you expect before working with master files.

1.2.4 Exporting metadata and generating access files

You can use the "Export metadata" button to save the currently displayed metadata to a PBCore XML sidecar file. Clicking the button will bring up a window where you can specify the folder and file name of the PBCore file, and where you can optionally embed existing XML from other software (such as OHMS) within a PBCore instantiationExtension element. In addition, it is possible to automatically run *lame* to generate an MP3 access file from the target BWF. Only VBR encoding is currently supported.

Note: The metadata exported to PBCore correspond to the texts visible in the autoBWF GUI at the time that the "Export metadata" button was clicked. Note that if you have edited metadata fields but have not saved the metadata to the BWF, then the exported PBCore will **not** match the internal metadata in the BWF.

1.2.5 How does autoBWF generate metadata?

From filenames

autoBWF was designed assuming that filenames follow a convention similar to that used by the Indiana University Archives of Traditional Music as described in the "Sound Directions" report. It uses a regular expression ("regex") to parse the filename and extract components (called "capture groups") for use in generating the Description and OriginationDate BWF fields. This regex is specified in the configuration JSON file, see *Program behavior*.

Specifically, autoBWF expects that the regex has capture groups for the following three pieces of data:

- an item-level identifier of the physical instantiation corresponding to this digital object in arbitrary format
- a file use indicator
- the date of file creation in hyphenless ISO 8601 format

These data must occur in that order within the filename. The file use indicator is translated into natural language using the "fileuse" element in the JSON configuration file, see *Program behavior*.

If autoBWF cannot parse the filename, then it will display a warning, use the OS file creation date and time to generate OriginationDate, OriginationTime, and OriginatorRef, and will leave Description blank.

If the naming convention at your archives is different enough from the above that it cannot be accomodated by modification of the regex, then that will require modifications to the Python codebase. If the modifications can be made without significant rewriting, then you may be able to convince the maintainer of the project into making changes to accomodate your needs. Please create an "issue" on GitHub that describes your needs (click on the green "New issue" button), and let's talk about it! (Please note that anything you write in a GitHub issue is visible to the entire Internet, so don't include anything that you don't want to reveal publically.) If you want to make substantial local modifications, feel free to fork the project.

From operating system metadata

The values of OriginationTime and OriginatorRef are generated by combining the file creation dates and times obtained from OS metadata together with default values in `autobwfconfig.json`. If there is a conflict between the OS metadata date and that in the filename, then the program will display a warning and will allow you to choose which one you want to use.

1.3 Configuration

Program configuration is stored as a JSON file named `autobwfconfig.json` in a directory appropriate to your operating system. If the file does not exist, then a "starter" file will be created for you. You can find the location of your configuration file by running:

```
autoBWF --config
```

This file should be edited using the text editing program of your choice in order to reflect your repository's usage. In particular, you can customize the values in all of the dropdown menus, set the boilerplate copyright texts, and save the models and serial numbers of your hardware for constructing the BWF CodingHistory element.

1.3.1 Originator and repository information

- Set the pre-filled value for the BWF Originator element:

  ```
  "originator": "Apocryphal St. U. Archives"
  ```

- Set the value of the RIFF IARL element:

  ```
  "iarl": "US, Apocryphal State University Archives and Special Colletions"
  ```

- Set the repository code used to construct the BWF OriginatorRef element:

  ```
  "repocode": "ApoSU"
  ```

1.3.2 Program behavior

- Control whether `bwfmetaedit` is sent the `--accept-nopadding` flag (no if 0, yes otherwise):

  ```
  "accept-nopadding": 1
  ```

- Specify the filename format (see below for full usage details):

  ```
  "filenameRegex": ".*_([0-9-]+_[A-Z]{2}\\\d+)_\\\d+_([a-zA-Z]+)_(\\\d+)\\\.wav"
  ```

- Control how the file use portion of the file name is translated into a file use statement in the BWF Description:

```
"fileuse": {
    "pres": "Preservation Master",
    "presInt": "Preservation Master-Intermediate",
    "prod": "Production Master"
}
```

1.3.3 Combo box elements

The prefilled values of the autoBWF combo box GUI elements (ISFT, ITCH, ISRC, Copyright owner, Form) are controlled by list value associated with the appropriate key:

```
"isft": [
    "Audacity 2.0.3.0",
    "Audacity 2.3.2"
],
"technician": [
    "Doe, John",
    "Doe, Jane"
],
"source": [
    "Billy Bob Apocryphal papers",
    "Apocryphal University Radio records"
],
"owner": [
    "Apocryphal State University",
    "Big Donor"
],
"form": [
    "Live sound recordings",
    "Interviews",
    "Radio programs",
    "Oral histories",
    "Personal recordings",
    "Field recordings",
    "Poetry readings (Sound recordings)"
]
```

If you want a combo box to be empty be default but still have a set of preconfigured choices in the dropdown, then make the first element of the list an empty string:

```
"technician": [
     "",
     "Doe, John",
     "Doe, Jane"
 ]
```

1.3.4 Copyright boilerplate

The copyright dropdown menu and associated texts are controled by the “copyright” key:

```
"copyright": {
    "list": ["Generic", "CC-BY-SA"],
    "Generic": "Publication and other forms of distribution (including online sharing␣
↪and streaming) may be restricted. For details, contact the Apocryphal State␣
↪University Archives.",
```

(continues on next page)

(continued from previous page)

```
    "CC-BY-SA": "This content is copyright by the Apocryphal State University, and is␣
↪licenced under Creative Commons BY-SA. See https://creativecommons.org/licenses/by-
↪sa/4.0/ for details."
}
```

Within it is the “list” key, the corresponding value of which is a list of user-friendly names for the various copyright choices in the form that they should appear in the dropdown menu. That should be followed by key-value pairs corresponding to each of those user-friendly names and the corresponding boilerplate text that should be filled in the text field. For example, we can add a “public domain” menu choice as follows:

```
"copyright": {
    "list": ["Generic", "CC-BY-SA", "Public domain"],
    "Generic": "Publication and other forms of distribution (including online sharing␣
↪and streaming) may be restricted. For details, contact the Apocryphal State␣
↪University Archives.",
    "CC-BY-SA": "This content is copyright by the Apocryphal State University, and is␣
↪licenced under Creative Commons BY-SA. See https://creativecommons.org/licenses/by-
↪sa/4.0/ for details.",
    "Public domain": "This content has been placed into the public domain."
}
```

1.3.5 CodingHistory constructor menus

autoBWF constructs the CodingHistory text based on user selections in the dropdown menus to the right of the text box. The contents of those menus is controlled by the following JSON elements:

```
"deck": {
    "list": ["Studer", "Realistic"],
    "Studer": "Studer A810 SN:11223344",
    "Realistic": "Realistic 909A SN:1234321"
},

"adc": {
    "list": ["Lynx"],
    "Lynx": "Lynx Aurora 16 SN:897969"
},

"software": {
    "list": ["Audacity - Mac", "Audacity - Linux"],
    "Audacity - Mac": "Audacity 2.0.3.0 (Mac)",
    "Audacity - Linux": "Audacity x.x.x. (Linux Ubuntu)"
},


"media": [
    "1/4 inch open reel",
    "cassette"
],

"speed": [
    "",
    "7.5 ips",
    "3.25 ips"
],
```

(continues on next page)

(continued from previous page)

```
"eq": [
    "",
    "Dolby B",
    "Dolby C"
],

"type": [
    "",
    "CrO2",
    "Metal"
],
```

Some of these elements have a "list" key similar to the copyright dropdown menu configuration ("deck", "adc", "software"), while for the remainder the text in the dropdown menu is the same as the text inserted into the CodingHistory (similar to the combo box configuration).

1.4 Auxilliary tools

The autoBWF package also provides scriptable command line tools to facilitate audio file and metadata management.

1.4.1 autosplice

Usage:

```
autosplice <EDL file>
```

where *<EDL file>* is a text file that is vaguely reminiscent of an edit decision list.

Specifically, the EDL file consists of any number of lines of the form

```
filename.wav <in-time> <out-time>
```

which may also optionally contain *fade <in duration> <out duration>* and/or *pad <start padding> <end padding>*. This will append the audio in *filename.wav* from time *<in-time>* to time *<out-time>* (relative to start of the file). All times and durations are in SoX time specification syntax (see the Sox man page for details). The optional *fade* and *pad* parameters add fade in and/or fade out (both values are required: set either to zero if no fade is desired), or padding with silence at beginning and/or end (both values are required: set either to zero if no padding is desired). Padding is added before fade in and after fade out.

The input file must end with a line containing only a filename, and optionally the text *contrast <value>*, but no in or out times. This specifies the name of the output file, as well as any optional SoX "contrast" (a form of audio dynamic range compression).

Example of an EDL input file:

```
file1.wav 02:40+54392s 03:03+26995s fade 0 5
file1.wav 03:31+26995s 04:35+08823s fade 0 20 pad 1.5 2.5
file2.wav 14:12+26995s 15:02+08823s fade 15 5
file3.wav 0 5
output_file.wav
```

Security note

`autosplice` calls `sox` using unsanitized strings passed to the Python `subprocess.call()` function with the `shell=True` argument. This is a theorectical security risk, as a maliciously-crafted EDL file obtained from an untrusted source could result in a shell injection attack. It is mitigated by the fact that such a malicious file would be easily detectable by simple visual inspection.

1.4.2 bwf2pbcore

Usage:

```
bwf2pbcore [-h] [--ohms OHMSFILE] infile [infile ...]
```

This is a CLI version of the PBCore export functionality provided by the "Export metadata" button of the autoBWF GUI. bwf2pbdore extracts the embeded BWF metadata in each <*infile*> and saves it as a PBCore XML sidecar file. The filename of the generated sidecar file will be the same as that of <*infile*> with "_pbcore" inserted just before the ".xml" extension (i.e. the input file "foobar.wav" will result in a sidecar file with the name "foobar_pbcore.xml")

bwf2pbcore checks for the presence of a file with the same name as the <*infile*>, but with "_ohms.xml" as a suffix and extension (i.e. for the input file "foobar.wav", it will look for the file "foobar_ohms.xml"). If such a file is found, it is assumed that it contains XML metadata exported from the Oral History Metadata Synchronizer, and the contents will be inserted into the PBCore XML within an instantiation-level <*extensionEmbedded*> element. Alternatively, the OHMS file can be specified using the `--ohms` option.

1.4.3 bwf2csv

Usage:

```
bwf2csv [-h] [-h] [--digest] [-o OUTFILE] infile [infile ...]
```

Selected elements of embeded BWF metadata in each <*infile*> will be extracted and output to *stdout* in CSV format (one line per BWF file). Multiple <infile>s can be given, or generated using a shell glob (e.g. **.wav*). The `-o` option can be used to specify the output CSV file, which is particularly useful when running bwf2csv through `find -exec` in UNIX-like environments. The `--digest` option can be used to verify the MD5 data chunk digests of the BWF file(s). Note that this will be slow if the Wave files are large.

1.4.4 autolame

Usage:

```
autolame [-h] [-o OUTFILE] [--vbr-level VBR_LEVEL] infile [infile ...]
```

This is a CLI version of the PBCore "Generate MP3" functionality provided by the "Export metadata" button of the autoBWF GUI.Each <*infile*> will be converted to mp3 and the result will be saved to the same file name with the extension changed to mp3. Multiple <infile>s can be given, or generated using a shell glob (e.g. **.wav*). Selected embedded metadata values from the BWF files are migrated to ID3v2 tags in the resulting mp3 files.

An output file name can be specified using the `-o` option, but in that case only one input file is allowed.

The default VBR level is currently 7.

CHAPTER 2

Known issues

- Reading of XMP data causes a temp file to be created and deleted in the same directory as the WAVE file. This may cause a change to the modification time for the directory, which could cause a problem for some digital preservation schemes.
- `autoBWF` strives to write valid XMP, but when reading Wave files it makes significant assumptions about the structure of the XMP XML beyond those mandated by the XMP standard. Therefore, it is only capable of reliably reading XMP generated by `autoBWF` itself, and metadata written by other software may not be correctly parsed.