



**Reading
App Builder**

**Using aeneas for
Audio-Text Synchronization**



Reading App Builder: Using aeneas for Audio-Text Synchronization

© 2017, SIL International

Last updated: 4 December 2017

You are free to print this manual for personal use and for training workshops.

The latest version is available at
<http://software.sil.org/readingappbuilder/resources/>

and on the Help menu of Reading App Builder.

Contents

1. Introduction	5
1.1. <i>What is aeneas?.....</i>	5
1.2. <i>Will it work with our language?</i>	6
1.3. <i>How long does it take?</i>	6
1.4. <i>How do we run aeneas?.....</i>	6
2. Windows Installation	7
3. Linux Installation	9
3.1. <i>Initial installation on Linux.....</i>	9
3.2. <i>Updating aeneas on Linux</i>	10
4. Using aeneas within Reading App Builder	10
4.1. <i>Running the Audio Synchronization wizard</i>	10
4.2. <i>Troubleshooting</i>	11
4.3. <i>Viewing and testing the timing files created by aeneas.....</i>	11
5. Fine Tune Timings.....	12
5.1. <i>Fine tune timing files using the Fine Tune Timings viewer</i>	12
5.2. <i>Fine tune timing files using Audacity</i>	14
6. Adding voices to eSpeak	14

Acknowledgements

Thank you very much to Alberto Pettarin, Italy, for all his work developing **aeneas** and for making it freely available for use in synchronising text and audio around the world.

Website: <https://github.com/readbeyond/aeneas/#aeneas>

These instructions were written using aeneas 1.7.2, released 09 mars 2017.

Thank you to Firat Özdemir for inventing the **Fine Tuning** feature, a modified version of which is built into Reading App Builder.

Website: <https://github.com/ozdefir/finetuneas>

Thank you to Daniel Bair for his work on the Windows installer for **aeneas**.

Website: <https://github.com/sillsdev/aeneas-installer>

1. Introduction

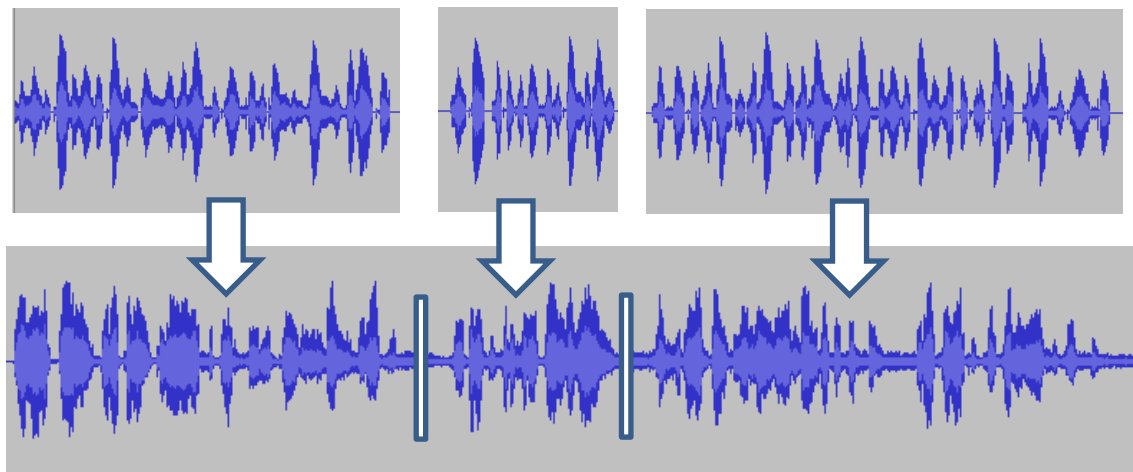
1.1. What is aeneas?

aeneas is a tool designed to “automagically synchronize audio and text”.¹ It takes an audio file and a list of phrases as inputs. Its output is a synchronization timing file, i.e. the same kind of labels file you would create manually in Audacity.

The way **aeneas** works is to synthesize each phrase using a text-to-speech engine, and then compare the synthesized audio files against the input audio file to find the phrase breaks. To do this, a lot of clever mathematical computation goes on behind the scenes:

Using the Sakoe-Chiba Band Dynamic Time Warping (DTW) algorithm to align the Mel-frequency cepstral coefficients (MFCCs) representation of the given (real) audio wave and the audio wave obtained by synthesizing the text fragments with a TTS engine, eventually mapping the computed alignment back onto the (real) time domain.²

Or to illustrate it more simply, the synthesized phrases...



...are mapped against the input audio file to find phrase breaks.

Best results are obtained:

- if you have a clearly spoken recording;
- if any background music is low enough in volume not to confuse the system;
- if the synthesized phrases correspond well enough to how the language is pronounced.

¹ <https://github.com/readbeyond/aeneas/#aeneas>

² <https://github.com/readbeyond/aeneas/#one-sentence-explanation-pro-edition>

1.2. Will it work with our language?

aeneas is producing impressively accurate timing files for languages around the world. We would recommend you try it out and see how well it works for your text and audio files.

What about special characters or tone marks? Since the text-to-speech engine copes best with simple A-Z Roman scripts, the **Synchronize using aeneas** wizard in Reading App Builder has a **Character Replacement** page, designed to replace special characters with simple ones (b to b, ε to e, etc.). It also strips out accents and tone marks. The comparison algorithms are tolerant enough to match up the phrases even though the pronunciation differs.

If you have a non-Roman script, you will need to define a set of character replacements, transliterating your non-Roman script into a simple A-Z Roman script (अ to a, क to ka, उ to u, etc.). This approach has already been used successfully to synchronize text and audio in Hebrew. If you create a set of additional character replacements that work well for your script and language, please let us know so that we can add them to the default set of replacements.

You can find a selection of Romanization tables here:

<http://www.loc.gov/catdir/cpsd/romansource.html>

1.3. How long does it take?

For a typical phrase-by-phrase synchronisation, running **aeneas** in **Windows** takes 1-2 minutes to process an hour of audio. Actual times will depend on the speed of your computer.

1.4. How do we run aeneas?

There are three ways to run aeneas:

1. Install **aeneas** on your computer and use the **Synchronize using aeneas wizard** in RAB to launch the synchronization process.
 - This is the most common way of running **aeneas**, and is especially useful for those who do not have a good internet connection. It will add the timing files to your RAB project as they are generated.
 - See section 2 for instructions on how to install **aeneas** on Windows, and section 3 on how to install **aeneas** on Linux. This is especially easy for Ubuntu users since **aeneas** is included automatically in the RAB installation.
2. Run aeneas online using the **aeneas** web app, <http://aeneasweb.org>. To do this, use the **Synchronize using aeneas wizard** in RAB to create a “job” zip file which you upload to the web app. You will receive the processed timing files by email.
 - This method is useful for users who have a good internet connection for uploading files.

- See <http://aeneasweb.org/help> for more information.
3. Install aeneas on a different computer, such as a Linux server, and use the **Synchronize using aeneas wizard** in RAB on your computer to create a “job” zip file which you can process on the other computer. When the process is completed, copy the timing files back to your computer.
 - This method is useful for those with slow computers who have a lot of audio to process. You could ask your IT department to set up a server for this purpose.
 - See the [aeneas documentation](#) on how to execute an aeneas job.

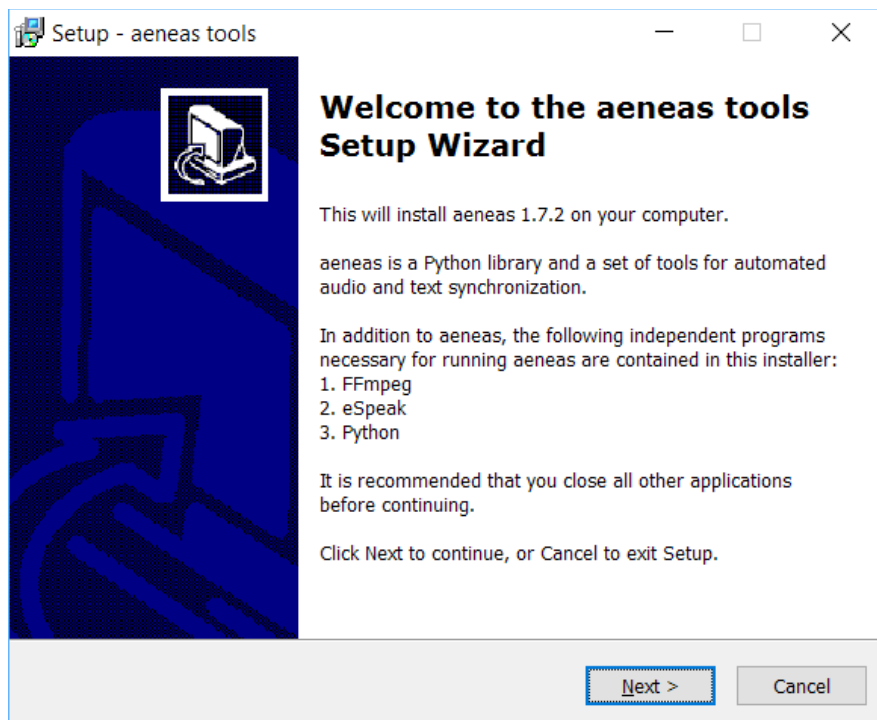
2. Windows Installation

To set up **aeneas** on Windows, there are several programs and modules to download and install: FFmpeg, eSpeak, Python and aeneas. These can all be installed with a single setup program.

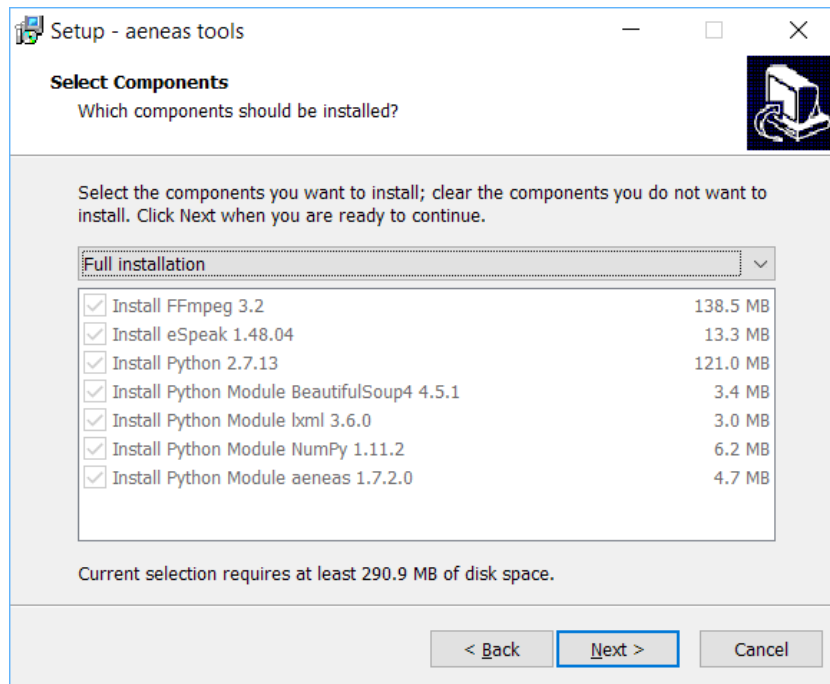
1. Go to the **Download** page on the Reading App Builder website <http://software.sil.org/readingappbuilder/download/> and download the latest aeneas setup program for Windows. You will find it under the heading **Audio Synchronization Tools**.

The filename will be something like **aeneas-windows-setup-1.7.2.exe**.

2. Double-click the file you have downloaded to start the installation wizard.



3. Follow the instructions in the wizard. You can accept all the defaults.
4. On the **Select Components** page, you need the **Full Installation** which should be selected by default.



5. On the **Ready to Install** page, click **Install**.

You will see the wizard running several different installers: for FFmpeg, eSpeak, Python. It will also install three Python modules.

Before the wizard completes, you should see a command box appear for a few seconds which verifies the aeneas installation.

```
C:\WINDOWS\system32\cmd.exe
[INFO] aeneas OK
[INFO] ffprobe OK
[INFO] ffmpeg OK
[INFO] espeak OK
[INFO] aeneas.tools OK
[INFO] shell encoding OK
[INFO] aeneas.cdtw COMPILED
[INFO] aeneas.cmfcc COMPILED
[INFO] aeneas.cew COMPILED
[INFO] All required dependencies are met and all available Python C extensions are compiled
[INFO] Enjoy running aeneas!
```

Important: If Reading App Builder was open while you installed aeneas, close it and start it again to ensure that it picks up the changes to your Path settings.

3. Linux Installation

3.1. Initial installation on Linux

Important:

If you are using Ubuntu 14.04 (Trusty) or later, then the **aeneas** software is a required dependency of Reading App Builder. This means that aeneas is automatically installed along with Reading App Builder and you do not have to install it manually.

For earlier releases of Ubuntu, please follow these installation instructions:

1. If you don't already have git installed:

```
$ sudo apt-get install git
```

2. Download the latest **aeneas** software:

```
$ git clone https://github.com/readbeyond/aeneas.git
```

3. Move to the **aeneas** folder:

```
$ cd aeneas
```

4. Install the **aeneas** dependencies:

```
$ sudo bash install_dependencies.sh
```

This will take some time to download all the different programs and modules, depending on the speed of your internet connection.

5. If there are problems with the FFmpeg install, a workaround is to download the **FFmpeg static build** from <http://johnvansickle.com/ffmpeg>. This is a file of the form **ffmpeg-git-64bit-static.tar.gz**. Unzip it and add links to the two binaries, **ffmpeg** and **ffprobe**:

```
$ sudo ln -s /path/to/ffmpeg/ffmpeg ffmpeg
$ sudo ln -s /path/to/ffmpeg/ffprobe ffprobe
```

where /path/to/ffmpeg is the folder to which you have unzipped the tar.gz file.

6. Compile the C extensions, used to run aeneas faster:

```
$ python setup.py build_ext --inplace
```

7. Check aeneas dependencies:

```
$ python aeneas_check_setup.py
```

Additional installation instructions for aeneas on Linux can be found here:

<https://github.com/readbeyond/aeneas/#installation>

3.2. Updating aeneas on Linux

If you are using Ubuntu 14.04 (Trusty) or later, then the **aeneas** software will be automatically updated when you update Reading App Builder.

For other Linux versions, to update the **aeneas** software to the latest version:

1. Open a terminal window and move to your **aeneas** folder.
2. Use git pull to retrieve the latest version from the repository:

```
$ git pull
```

3. Recompile the C extensions:

```
$ python setup.py build_ext --inplace
```

4. Check aeneas dependencies:

```
$ python aeneas_check_setup.py
```

4. Using aeneas within Reading App Builder

4.1. Running the Audio Synchronization wizard

You can run the **Audio Synchronization using aeneas** wizard from four places within Reading App Builder:

1. The **Audio Files** tab for a book or book collection. Select one or more rows in the table, right-click and select **Synchronize using aeneas**.
2. The **Audio Synchronization** tab for a book. Click on the button **Synchronize using aeneas**.
3. The **Book Collection** tab for a book collection. Right-click on a book and select **Synchronize using aeneas**.

4. The Apps tree on the left of the screen. Right-click on a book or book collection and select **Synchronize using aeneas**.

Follow the instructions in the wizard to configure the synchronization.

For Windows users, if the wizard asks, “Where have you installed aeneas?”, please follow the installation instructions in section 2 of this document.

After the final page of the wizard, the synchronization process will start in a separate command Window. It runs a batch file (or bash script) which calls an **aeneas** task for each chapter of each book.

As timing files are generated you will see their filenames being added to the table on the **Audio Files** page.

4.2. Troubleshooting

If the synchronization fails, please ensure that aeneas has been installed correctly. You can check the installation by selecting **Tools** ➤ **Check aeneas Installation...** from the Reading App Builder main menu.

A common problem on Windows is that **eSpeak** stops working after a while. To correct this, run the aeneas Windows setup program again.

To verify that eSpeak has been installed correctly and is in your path, open a new Windows command prompt and type:

```
espeak hello
```

If all is well, you should hear the word “hello” pronounced.

4.3. Viewing and testing the timing files created by aeneas

To view a generated timing file, select a row on the **Audio Files** page, right-click, and select **View Timing File**.

To test the generated timing files with the text and audio, select a row in the **Audio Files** page, right-click and select **Export to HTML**.

5. Fine Tune Timings

5.1. Fine tune timing files using the Fine Tune Timings viewer

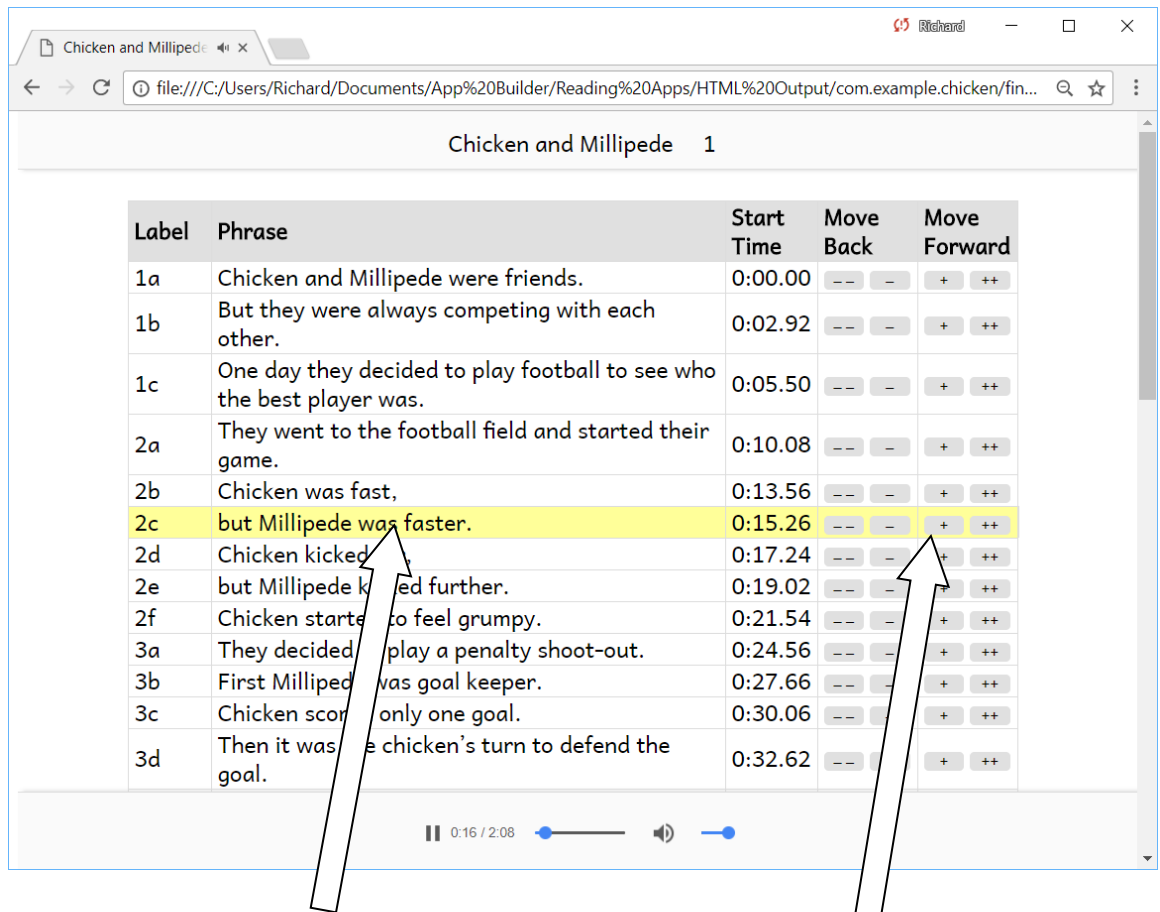
To fine tune a generated timing file:

1. Select a row on the Audio Files page, right-click, and select **Fine Tune Timings**.

A page should open in your default browser, looking something like this:

Label	Phrase	Start Time	Move Back	Move Forward
1a	Chicken and Millipede were friends.	0:00.00	-- -	+ ++
1b	But they were always competing with each other.	0:02.92	-- -	+ ++
1c	One day they decided to play football to see who the best player was.	0:05.50	-- -	+ ++
2a	They went to the football field and started their game.	0:10.08	-- -	+ ++
2b	Chicken was fast,	0:13.56	-- -	+ ++
2c	but Millipede was faster.	0:15.26	-- -	+ ++
2d	Chicken kicked far,	0:17.24	-- -	+ ++
2e	but Millipede kicked further.	0:19.02	-- -	+ ++
2f	Chicken started to feel grumpy.	0:21.54	-- -	+ ++
3a	They decided to play a penalty shoot-out.	0:24.56	-- -	+ ++
3b	First Millipede was goal keeper.	0:27.66	-- -	+ ++
3c	Chicken scored only one goal.	0:30.06	-- -	+ ++
3d	Then it was the chicken's turn to defend the goal.	0:32.62	-- -	+ ++

2. Click on the first row of the table to start the audio playing.
3. As soon as you hear the first word or two from the selected row, click on the next row.
4. Continue clicking on each row to listen for enough time to determine whether the timing break has been placed in the right place, i.e. not too early and not too late.
5. If you hear that the timing for a row needs to be fine-tuned, click the plus (+) button to move the start time forward by 0.1 seconds, the double-plus (++) to move forward by 1 second, the minus (-) button to move the start time backwards by 0.1 seconds, or the double-minus (--) to move back 1 second.



Click on each row for enough time to check if the audio starts at the right place.

Adjust the start time of the currently selected timing using the plus and minus buttons.

- When you are happy with the current row, continue clicking on the subsequent rows in the same way, fine tuning those that need adjusting.
- When you have reached the last row of the table, save your changes by clicking on the button **Save Changes to Timing File**.

Important: For security reasons, your browser does not have permission to save the modified timing file in its original location. It will be saved to your default Downloads folder. Reading App Builder can watch this folder for modified files and move them back to your project data folder. Specify the Downloads folder to watch in **Settings** ➤ **Files**. Otherwise, you will need to move the modified files back to your project manually, e.g. using the **Add Timing Files** button.

5.2. Fine tune timing files using Audacity

Instead of using the Fine Tune Timings viewer, you can make corrections in Audacity. To do this:

1. Open the audio MP3 file in Audacity.
2. Import the timing file, by selecting **File** ➤ **Import** ➤ **Labels...**
3. Zoom into the waveform.
4. Adjust the position of any labels that are not in the right place.
5. Export the modified labels track, using **File** > **Export Labels...**
6. Test the synchronisation again within Reading App Builder using the **Export to HTML** utility (see above) to see if the timing problems have been resolved.

6. Adding voices to eSpeak

By using character replacements in the **Synchronize using aeneas** wizard, you will find that you can get good synchronization results for many languages using the existing eSpeak voices.

Adding a new voice in eSpeak is not easy, but if you would like to try, please follow the instructions here: http://espeak.sourceforge.net/add_language.html