High-quality audio (albums, music for listening, film scores, etc.), including CD quality and HD quality audio, use 16-bit (CD quality) or 24-bit (HD quality) sample resolution at a 44.1 kHz or a 48 kHz sampling rate.

Ultra-high-definition audio, or UHD Audio, uses a 24/96 configuration, or a 24-bit resolution with a 96 kHz sample rate.

The best way to show this work process is by using a hands-on approach, so let's get right into the optimization process!

Digital Audio Optimization: Work Process

The logical work process for optimizing a digital audio sample asset across hardware devices and using widely adopted open source content delivery platforms involves creating a 32-bit asset in Audacity and using a 44.1 kHz or a 48 kHz sampling rate, and then creating a baseline 16-bit uncompressed PCM format. You target 16-bit audio in this chapter, because all Android OS, Kindle, and HTML5 OS devices support 16-bit audio. If you're targeting HD audio platforms and devices, you can perform this work process using 24-bit audio.

Baseline PCM File: Something to Measure Against

The first thing that you want to do is establish the "baseline" file size that you will measure your asset compression against using each of the major digital audio file formats. Fortunately, there's an uncompressed file format called pulse-code modulation (PCM) that serves this purpose and provides you with an additional widely supported digital audio file format to use, if you need it, on any of the open publishing platforms (all of which will support PCM).

There is another important rationalization for exporting an uncompressed PCM baseline file, besides measuring compression against. When any of your compressed (lossy or lossless) assets are decompressed (decoded) into system memory, PCM represents a **memory footprint** that is needed to hold and play back the digital audio asset. So from a publishing and programming standpoint, you should also want to know how much **device memory** is needed to hold your application's digital audio assets. This should be calculated by adding up all of your PCM file sizes.

Open your **CH9.aud** project file and the **Stereo Track** drop-down menu, as shown in Figure 12-1. Select **Set Sample Format** and set it to 16-bit PCM, or 24-bit PCM if you're targeting HD Audio.

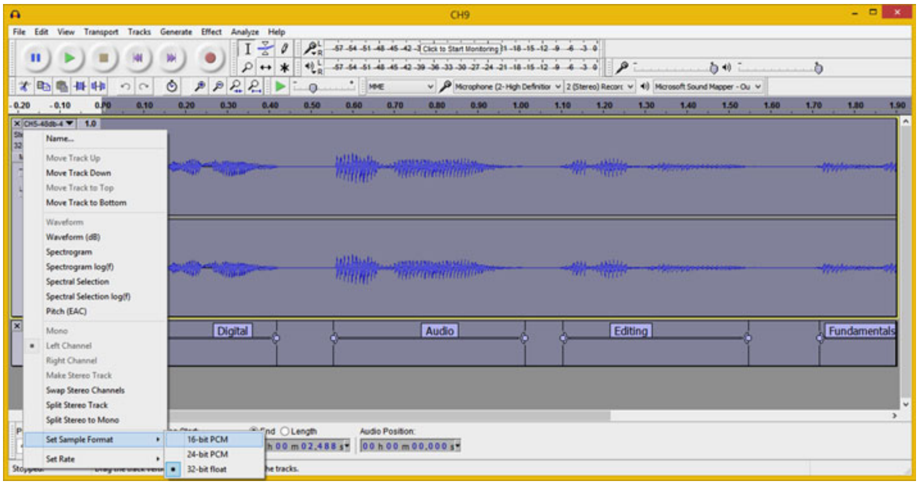


Figure 12-1. *Set Sample Format to 16-bit with Stereo Track menu*

The next step is to use the Audacity **File ➤ Export Audio** menu sequence to export a **WAV** file if you are on Windows, or an **AIFF** file if you are on Mac OS X (see Figure 12-2).

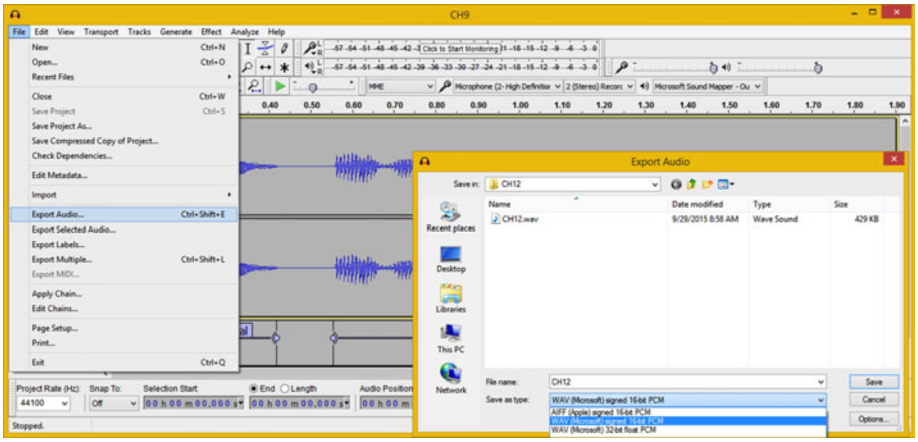
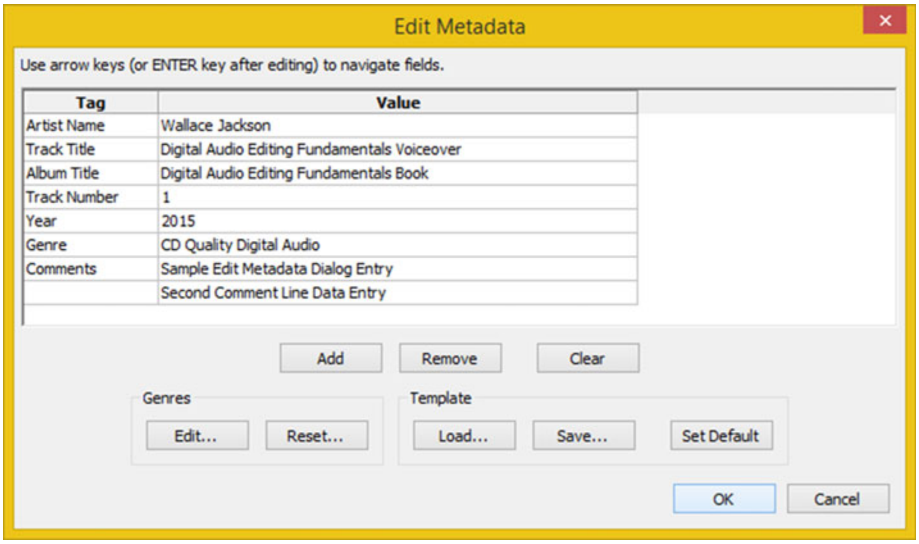


Figure 12-2. *Export baseline 16-bit PCM WAV or AIFF file format*

Once you click the **Save** button, the **Edit Metadata** dialog appears for you to enter any metadata that you want included in your audio file (see Figure 12-3). The data is stored in each file format along with the audio data and it can be accessed in Android. This dialog appears for each audio format's export.


Edit Metadata

Use arrow keys (or ENTER key after editing) to navigate fields.

Tag	Value
Artist Name	Wallace Jackson
Track Title	Digital Audio Editing Fundamentals Voiceover
Album Title	Digital Audio Editing Fundamentals Book
Track Number	1
Year	2015
Genre	CD Quality Digital Audio
Comments	Sample Edit Metadata Dialog Entry
	Second Comment Line Data Entry

Figure 12-3. *The Audacity 2.1.1 Edit Metadata dialog*

If you are wondering how your applications will read and support audio metadata if you choose to install the data in the audio file, Android Studio does support metadata retrieval with its **MediaMetadataRetriever** class, which developers utilize for this very specific purpose.

If for some reason your Android application needs to use audio media metadata, you should use the Edit Metadata dialog along with the Android `MediaMetadataRetriever` class, which you can learn more about at `developer.android.com/reference/android/media/MediaMetadataRetriever.html`.

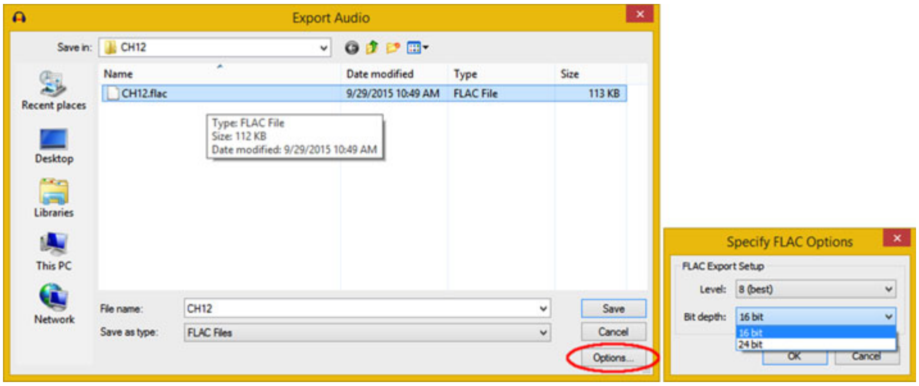
If you look at the `CH12.wav` 16-bit PCM file that you've saved out, you see that the file size is 428 KB. So your baseline uncompressed data and system memory footprint for a 2.48-second voice-over sample is 428 KB. You can use this number to determine the amount of compression that you will get with all the popular digital audio file formats you looked at in Chapter 4.

The next step is to go through the optimization process; that is, apply codec compression to your uncompressed (PCM) data using the different codecs supported in Android, Kindle, and HTML5. I am going to cover all the popular, open, digital audio formats.

Once that work process is completed, you are able to ascertain which resulting digital audio assets provide you with the highest quality digital audio playback in conjunctions with the lowest possible data footprint. Let's get right to it!

Exporting Lossless Audio: FLAC Audio Format

The first format that I am going to try out is the FLAC audio codec, because it uses lossless compression. This gives you a good idea of what kind of data footprint reduction you can get using compression, which does not throw away any of the original audio data. Using a lossless compression algorithm gives you as perfect a result as the 16-bit PCM WAV audio does! To do this, you'll again use the **File ➤ Export Audio** menu sequence. This time, you'll open the **Save as type** drop-down menu and select the **FLAC Files** format (see Figure 12-4).



CH12.flac | 9/29/2015 10:49 AM | FLAC File | 113 KB

Type: FLAC File
Size: 112 KB
Date modified: 9/29/2015 10:49 AM

File name: CH12
Save as type: FLAC Files

FLAC Export Setup
Level: 8 (best)
Bit depth: 16 bit
16 bit
24 bit

Figure 12-4. *Export Audio as FLAC format, in 16-bit at Level 8*

I named the file **CH12.flac** and saved it into my CH12 folder for this book. Notice that there are only FLAC files listed in the center area of the dialog. This is because now that you have selected the FLAC file format, this region in the dialog will only show FLAC files, and currently, there are none in the CH12 folder. Because I wanted to show file size for each format in Figure 12-4, I saved the file prior, so the size would be included in the screenshot, as well as the dialog settings and the options dialog (this way, I'm using only one screenshot rather than three).

To set the FLAC codec options, click the **Options** button and set the quality **Level** to 8 (best) and the **Bit depth** to use 16-bit data resolution. Note under the **Bit depth** drop-down that you can also use this FLAC codec for lossless, 24-bit HD audio. (I placed the **Export Audio** and **Specify FLAC Options** dialogs in one unified screen to save on the number of screenshots that I'm using.)

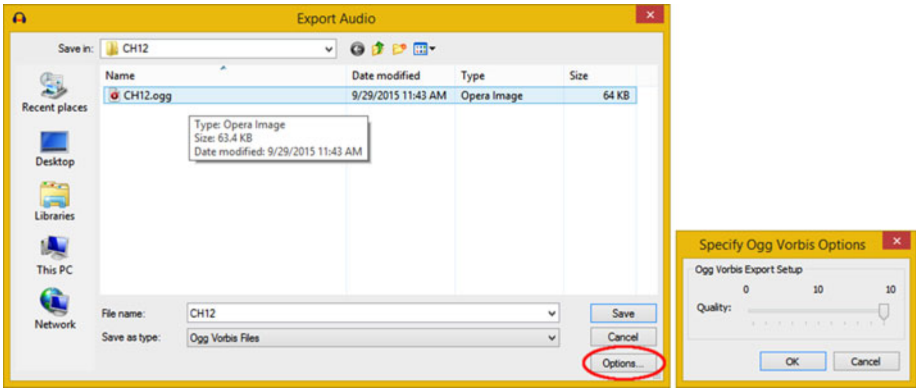
Once you've output the CH12.flac digital audio asset, go into the file manager to look at the file size, or use the **File ➤ Export Audio** menu sequence and mouse over **CH12.flac**.

You see that it is 112 KB, reduced by more than 382%; 112 ÷ 428 = 0.261682, or only 26% as large as the PCM file was, or 100% – 26% = 74% less data. The 1/x (inverse) function on your calculator of 0.261682 gives you 3.82142857, an impressive 382.142857% data footprint reduction, with zero loss of data (quality). The FLAC format is good for use in Android Studio applications development, in conjunction with the PNG image format.

Next, let's take a look at the other open source digital audio file format, Ogg Vorbis, to see if it can provide an even smaller data footprint. Since Ogg Vorbis is a lossy file format, it should give you an even smaller file size than the FLAC codec.

Exporting Lossy Audio: Ogg Vorbis Audio Format

Again, use a **File ➤ Export Audio** work process to open the Export Audio dialog. Select the **Ogg Vorbis Files** option from the **Save as type** drop-down menu. I named the file **CH12**, which produces a CH12.ogg file name, and put it into the CH12 folder (see Figure 12-5). Click the **Options** button and select a **Quality** setting level between 0 and 10. I used the maximum setting of 10 to start with. During an actual data footprint optimization session, you will probably try several settings to see how the data footprint–to-quality trade-off is affected by the Quality slider setting.



Save in: CH12

Name	Date modified	Type	Size
CH12.ogg	9/29/2015 11:43 AM	Opera Image	64 KB

Type: Opera Image
Size: 63.4 KB
Date modified: 9/29/2015 11:43 AM

File name: CH12
Save as type: Ogg Vorbis Files

Quality: 10

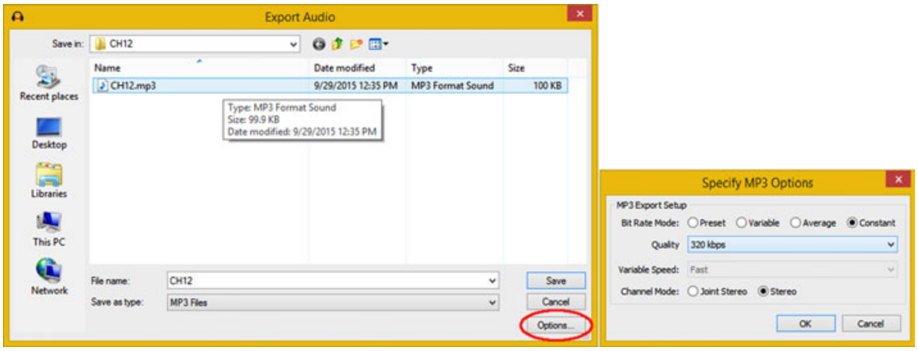
Figure 12-5. Export Audio in Ogg Vorbis format at Quality of 10

Once you have output a CH12.ogg audio asset, take a look at its file size. You see that it is 64 KB, reduced by 669% in file size; 64 KB ÷ 428KB = 0.14953, which is 15% of your baseline 428 KB. The inverse of this is 6.6875, or a 669% data reduction. This OGG file is 85% smaller than your PCM file.

Next, let's look at MP3, which is the most common lossy digital audio file format currently supported across publishing platforms. It should be quite interesting to see if MP3 can give you a smaller data footprint than the Ogg Vorbis codec.

Exporting Lossy MP3 Audio: MPEG3 Audio Format

Use you **File ➤ Export Audio** menu sequence again to bring up the Export Audio dialog. Set the **Save as type** drop-down selector to **MP3 Files** (see Figure 12-6). I named the file **CH12** and selected the **CH12 folder**. Then I clicked the **Options** button to open a **Specify MP3 Options** dialog (shown on the right in Figure 12-6). I used the maximum **Quality** bit-rate setting of 320 kbps, which is a high-quality setting for digital audio data. I am trying to get a high-end baseline for MP3 compression and still compare what MP3 can do against an uncompressed PCM baseline. I also selected the **Constant Bit Rate Mode**, because it is the easiest to decode, and a **Stereo Channel Mode**, since the file is currently in a Stereo Track configuration.



Save in: CH12

Name	Date modified	Type	Size
CH12.mp3	9/29/2015 12:35 PM	MP3 Format Sound	100 KB

Type: MP3 Format Sound
Size: 99.9 KB
Date modified: 9/29/2015 12:35 PM

File name: CH12
Save as type: MP3 Files

Bit Rate Mode: Preset Variable Average [✓ Constant]
Quality: 320 kbps
Variable Speed: Fast
Channel Mode: Joint Stereo [✓ Stereo]

Figure 12-6. *Export Audio in MP3 format at Quality of 320 kbps*

If you like, you can also try different Quality Bit Rate settings and the Variable and Average Bit Rate Modes to see how they affect the MP3 data footprint. If you decide to do this, differentiate your files from each other; simply name the file with the settings type in the file name. For example, a file with a 320 kbps Quality setting and a Variable Bit Rate Mode should be named **CH12_320kbps_vbr.mp3**. This way, you can compare the audio file size of the MP3 and any other codec file format and do the simple math to figure out the percentage of the data footprint reduction relative to the different settings. You'll do this next for the CH12_320kbps_cbr.mp3 file.

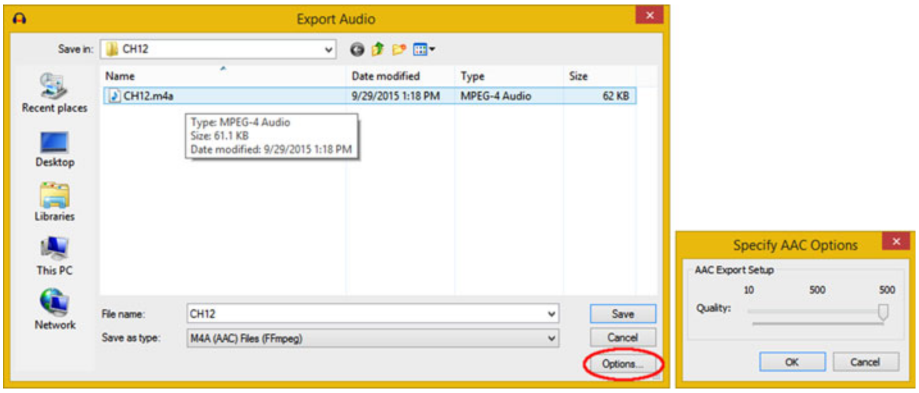
The CH12.mp3 file size is 100 KB, representing a 428% data footprint reduction. To calculate this, 100 KB ÷ 428 KB = 0.233644859, which is 23% of the original, uncompressed file size; 100% – 23% = 77% file size reduction.

If you use your 1/x (inversion) key on a calculator, you can get a percentage reduction coming from the other direction by inverting 0.2336449, which gives you 4.28. This means that you reduced the file size by 4.28 times, which equates to a 428% reduction.

Now that you have seen that your MP3 file size is larger than the Ogg Vorbis, let's see how the MPEG4 AAC in M4A file format data compression can improve your file's size-to-quality ratio compared to MPEG3. Since MPEG4 is more recent and uses a more advanced codec algorithm, M4A should provide a better file size–to-quality optimization ratio than either Ogg Vorbis or MP3 codecs.

Exporting Lossy M4A Audio: MPEG4 Audio Format

Follow the usual **File ➤ Export Audio** work process to invoke the Export Audio dialog and select **M4A (AAC) Files (FFmpeg)** from the **Save as type** drop-down menu. If you did not install the LAME and FFMPEG libraries from Chapter 1, the file formats covered from here on out will not be available to you in Audacity 2.1. As usual, name the file **CH12**, which is named **CH12.m4a** by the Audacity Export Audio dialog after you click the **Options** button and open the **Specify AAC Options** dialog (see Figure 12-7). Click **OK** to set the maximum **Quality** of 500, and then click **Save** to save it in the CH12 directory. I'm betting that even using a Quality setting of 500, the MPEG-4 AAC codec is so advanced that it will give you a smaller file size than Ogg Vorbis did at 64 KB.



CH12.m4a 9/29/2015 1:18 PM MPEG-4 Audio 62 KB

Type: MPEG-4 Audio
Size: 61.1 KB
Date modified: 9/29/2015 1:18 PM

File name: CH12
Save as type: M4A (AAC) Files (FFmpeg)

Quality: 500

Figure 12-7. *Export Audio in M4A AAC format; set Quality to 500*

Your CH12.m4a file size is 62 KB, representing 14% of the uncompressed data footprint, giving an 86% reduction. To figure this out, 62 ÷ 428 is 0.1449, or 13% of your original, uncompressed file size; 100% – 14% = 86% file size reduction. Invert the 0.1449 and you get 6.9, or an impressive 690% data footprint reduction.

Now that you have seen that your M4A AAC file size is the most impressive data footprint reduction thus far, let's see if the much more specialized AMR-NB (Narrowband) data compression codec will give you any further data footprint improvements over MPEG4 AAC.

Since this codec is optimized for voice, chances are the results are going to be significantly better than any that you have encountered thus far—so hold onto your hat! One thing to note about recording voices is that it does not require stereo.

Exporting Narrowband Audio: MPEG4 AMR Format

Even though the MPEG4 AMR-NB (Adaptive Multi-Rate Narrowband) codec and data format was originally designed (optimized) specifically for use with voice recording applications. However, there may be some other applications, for example, short-burst sound effects, that might obtain reasonable, if not fantastic, results by using this codec. As you know, any codec is simply a complex, mathematical algorithm, implemented using software. It doesn't discriminate, so, the only way to really find out which codec will give you the best compression to quality result with any given asset is to run the original uncompressed audio data through the codec, and then see what happens, which is what you are seeing over the course of this chapter.

AMR-NB is supported in Kindle, Android OS, Tizen OS, iOS and popular open HTML5 OS platforms and HTML5 browser software, including Mozilla Firefox and Google Chrome. It is also used in Blackberry OS and QuickTime.

Follow the usual **File ➤ Export Audio** work process to invoke the Audacity Export File dialog. Select your **AMR (narrow band) Files (FFmpeg)** option from the **Save as type** drop-down menu. As usual, name the file **CH12**, which is named CH12.amr by the Exporter, after you click **Save**.