

HTTP Live Streaming as a Secure Streaming Method

*Bobby Kania
Luke Gusukuma
Client: Keith Gilbertson
VT CS 4624 Semester Project
4/29/12*

Table of Contents

Abstract	3
What is HTTP Live Streaming?.....	4
Users' Guide	5
Requirements	5
How to Play Music.....	5
Developer's Guide to Using HTTP Live Streaming.....	6
Downloading HTTP Live Streaming	6
Using HTTP Live Streaming to Convert File	8
Setting up the Web Interface.....	12
Testing.....	13
Lessons Learned.....	14
Timeline/Schedule	14
Problems	14
Solutions	15
For our final solution we decided to use Apple's HTTP Live Streaming.	15
Requirements Addressed:	15
Acknowledgements.....	16
Client.....	16
Other	16
References.....	17

Abstract

The main purpose of this project was to find a secure streaming solution for audio files within the VT Library, specifically regarding the recital collection audio files that were donated by the Department of Music. Within the context of this project, the definition of secure streaming is a method of online streaming that ensures that copyrighted audio files are streamed while respecting copyright law.

The practical application of solving this problem has partially to do with advancement in mobile and internet technology as well as copyright issues (as stated above). In regards to mobile technology, many mobile devices currently being used are able to stream internet content such as audio and video, provided that the content is streamed efficiently. With regards to the internet technology, there has been increasing push towards using HTML5. Also with regards to internet technology, there has been a push for going away from Real Time Streaming Protocol (RTSP) due to internet security issues.

After researching several methods that are used for streaming media content on the web, including things such as URL hashing, Kaltura, a Linux implementation of HTTP Live Streaming, etc, we decided to use Apple's HTTP Live Streaming.

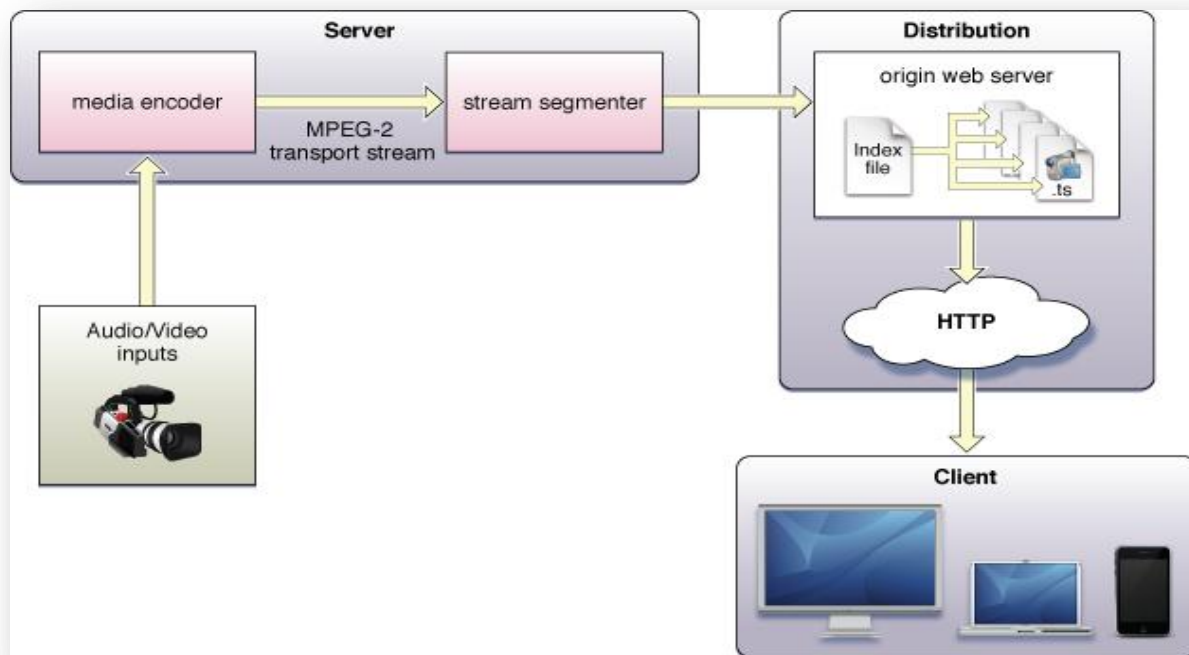
Apple's HTTP Live Streaming, which favors Apple products over other products, is both a practical and easy to implement solution for the library staff. In this document, we will go over three main points. The first point is the implementation of HTTP Live Streaming which includes the use of HTTP Live Streaming with regards to client side use, server side use, and the installation utility software. The second point is our procedure and project timeline. The third point will be the main problems that we confronted regarding both inherent issues (such as copyright issues, efficiency, etc) as well as implementation and research issues (such as system requirements and false leads in research).

What is HTTP Live Streaming?

HTTP Live Streaming lets you stream audio and video using the http protocol. The way HTTP Live Streaming works is it first takes your audio or video and sends it through a media encoder, and then sends it through a stream segmenter which gives you a set of TS (AAC files if converting an AAC file) files and a corresponding index which you can store on your server to distribute to users.

The media encoder takes your audio or video file and essentially turns it into the audio file format that you want to use for delivery. Once that's done, the stream segmenter chops your audio file into several segments that are saved as TS files (transport stream files) with an index file which dictates the order of the TS files. The TS files and the index file is what you put on your server to distribute for streaming.

The client software begins by fetching the index file based on an identifying URL for the stream. Then the client software uses the index file to process a TS files into a single continuous stream for the user to listen to/watch.



Users' Guide

Requirements

If you are building a server that uses HTTP Live Streaming, you need to be running Mac OS X 10.6 or later.

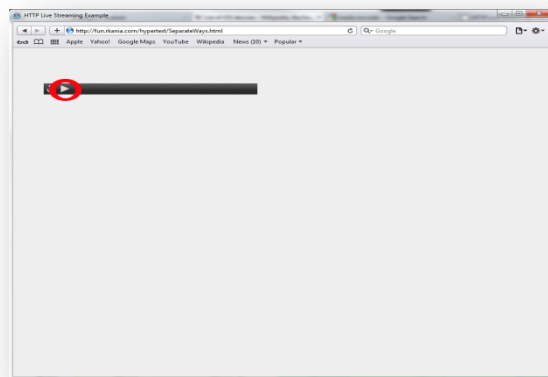
Users who are trying to stream content on your server using HTTP Live Streaming via a computer will need to use Mac OS X 10.6 as their operating system and Safari as their web browser.

Users of the mobile devices iPhone, iPod Touch, and iPad can also stream content using HTTP Live Streaming.

- Server side requirements (to use Apple's HTTP Live Streaming tools)
 - Mac running Mac OS X 10.6 or later
- Distribution server
 - Tested successfully on Apache
- User side requirements
 - Computer requirements
 - Mac running Mac OS X 10.6 or later
 - Web browser
 - Safari
 - Mobile devices
 - iPhone, iPod Touch, iPad

How to Play Music

Open your Safari browser and navigate to the site where your audio is hosted and press the play button (circled in red).

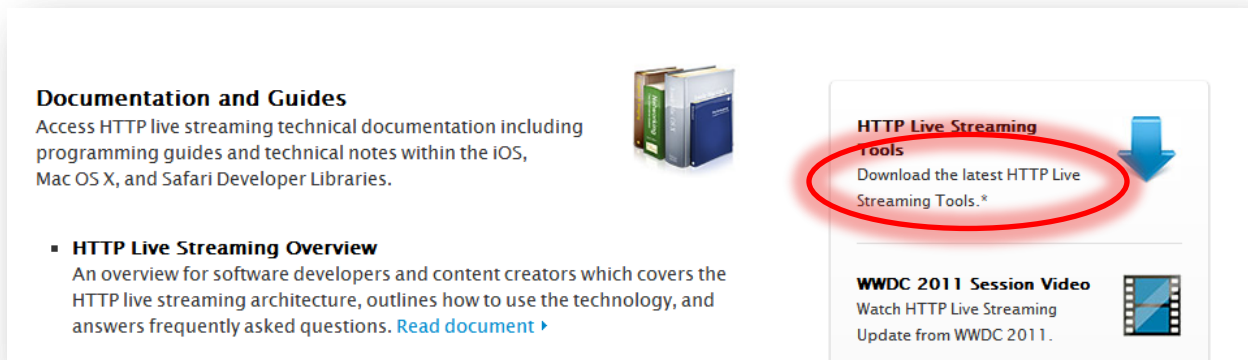


Developer's Guide to Using HTTP Live Streaming

Requirements: You must be using a Mac running Mac OS X 10.6 or later.

Downloading HTTP Live Streaming

1. Go to <https://developer.apple.com/resources/http-streaming/>
2. Click on “Download the latest HTTP Live Streaming Tools” on the right sidebar



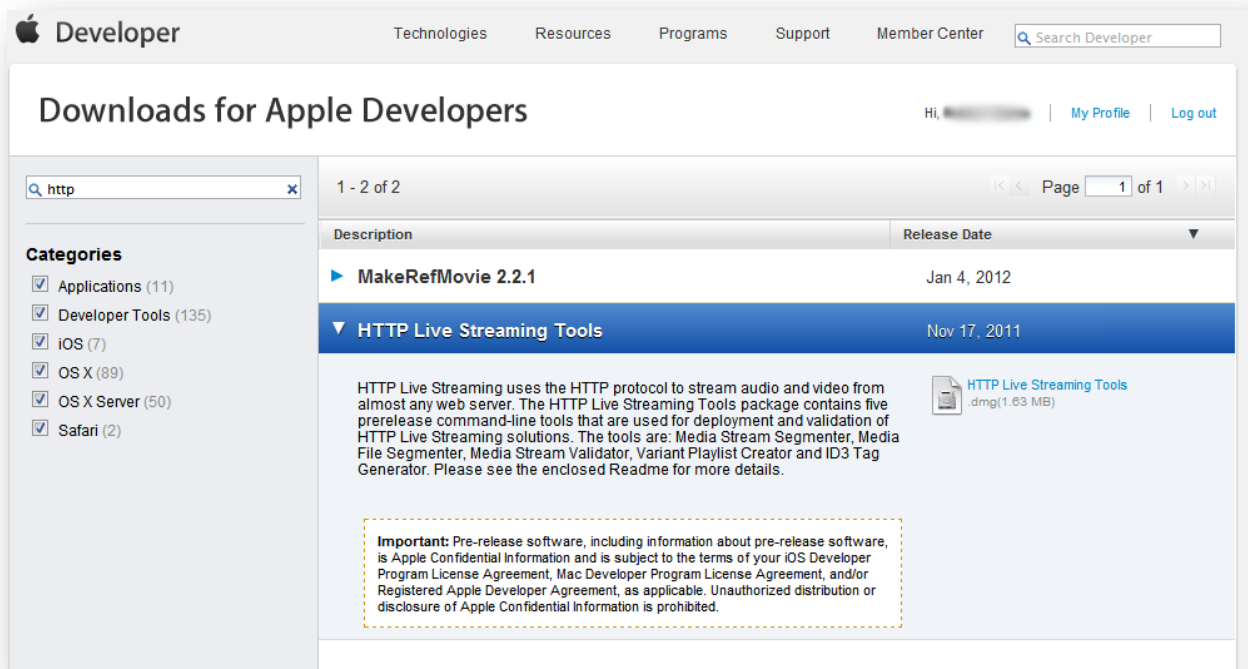
Documentation and Guides
Access HTTP live streaming technical documentation including programming guides and technical notes within the iOS, Mac OS X, and Safari Developer Libraries.

- **HTTP Live Streaming Overview**
An overview for software developers and content creators which covers the HTTP live streaming architecture, outlines how to use the technology, and answers frequently asked questions. [Read document](#) ▶

HTTP Live Streaming Tools
Download the latest HTTP Live Streaming Tools.*

WWDC 2011 Session Video
Watch HTTP Live Streaming Update from WWDC 2011.

3. Log in with your Apple Developer ID
4. If you have the correct permissions, you should be able to see “HTTP Live Streaming Tools” as a download option.



Apple Developer Technologies Resources Programs Support Member Center Search Developer

Downloads for Apple Developers

Hi, [Name] | My Profile | Log out

1 - 2 of 2 Page 1 of 1

Description	Release Date
▶ MakeRefMovie 2.2.1	Jan 4, 2012
▼ HTTP Live Streaming Tools	Nov 17, 2011

HTTP Live Streaming uses the HTTP protocol to stream audio and video from almost any web server. The HTTP Live Streaming Tools package contains five prerelease command-line tools that are used for deployment and validation of HTTP Live Streaming solutions. The tools are: Media Stream Segmenter, Media File Segmenter, Media Stream Validator, Variant Playlist Creator and ID3 Tag Generator. Please see the enclosed Readme for more details.

Important: Pre-release software, including information about pre-release software, is Apple Confidential Information and is subject to the terms of your iOS Developer Program License Agreement, Mac Developer Program License Agreement, and/or Registered Apple Developer Agreement, as applicable. Unauthorized distribution or disclosure of Apple Confidential Information is prohibited.

[HTTP Live Streaming Tools.dmg \(1.63 MB\)](#)

5. Click on the link on the right hand side labeled “HTTP Live Streaming Tools.”

The screenshot shows the Apple Developer website interface. At the top, there is a navigation bar with links for Technologies, Resources, Programs, Support, and Member Center, along with a search bar. Below this is the main heading 'Downloads for Apple Developers'. A search bar on the left contains the text 'http'. The search results are displayed in a table with columns for 'Description' and 'Release Date'. The first result is 'MakeRefMovie 2.2.1' with a release date of 'Jan 4, 2012'. The second result is 'HTTP Live Streaming Tools' with a release date of 'Nov 17, 2011'. This second result is highlighted in blue. Below the table, there is a description of the 'HTTP Live Streaming Tools' package, which includes five prerelease command-line tools. A red circle highlights the download link 'HTTP Live Streaming Tools.dmg (1.63 MB)'.

Description	Release Date
▶ MakeRefMovie 2.2.1	Jan 4, 2012
▼ HTTP Live Streaming Tools	Nov 17, 2011

HTTP Live Streaming uses the HTTP protocol to stream audio and video from almost any web server. The HTTP Live Streaming Tools package contains five prerelease command-line tools that are used for deployment and validation of HTTP Live Streaming solutions. The tools are: Media Stream Segmenter, Media File Segmenter, Media Stream Validator, Variant Playlist Creator and ID3 Tag Generator. Please see the enclosed Readme for more details.

Important: Pre-release software, including information about pre-release software, is Apple Confidential Information and is subject to the terms of your iOS Developer Program License Agreement, Mac Developer Program License Agreement, and/or Registered Apple Developer Agreement, as applicable. Unauthorized distribution or disclosure of Apple Confidential Information is prohibited.

6. This will download the HTTP Live Streaming toolkit to your computer. You will now need to open this download and install the .pkg file. At the time of this writing this was labeled “Streaming Tools Beta – 138.pkg”

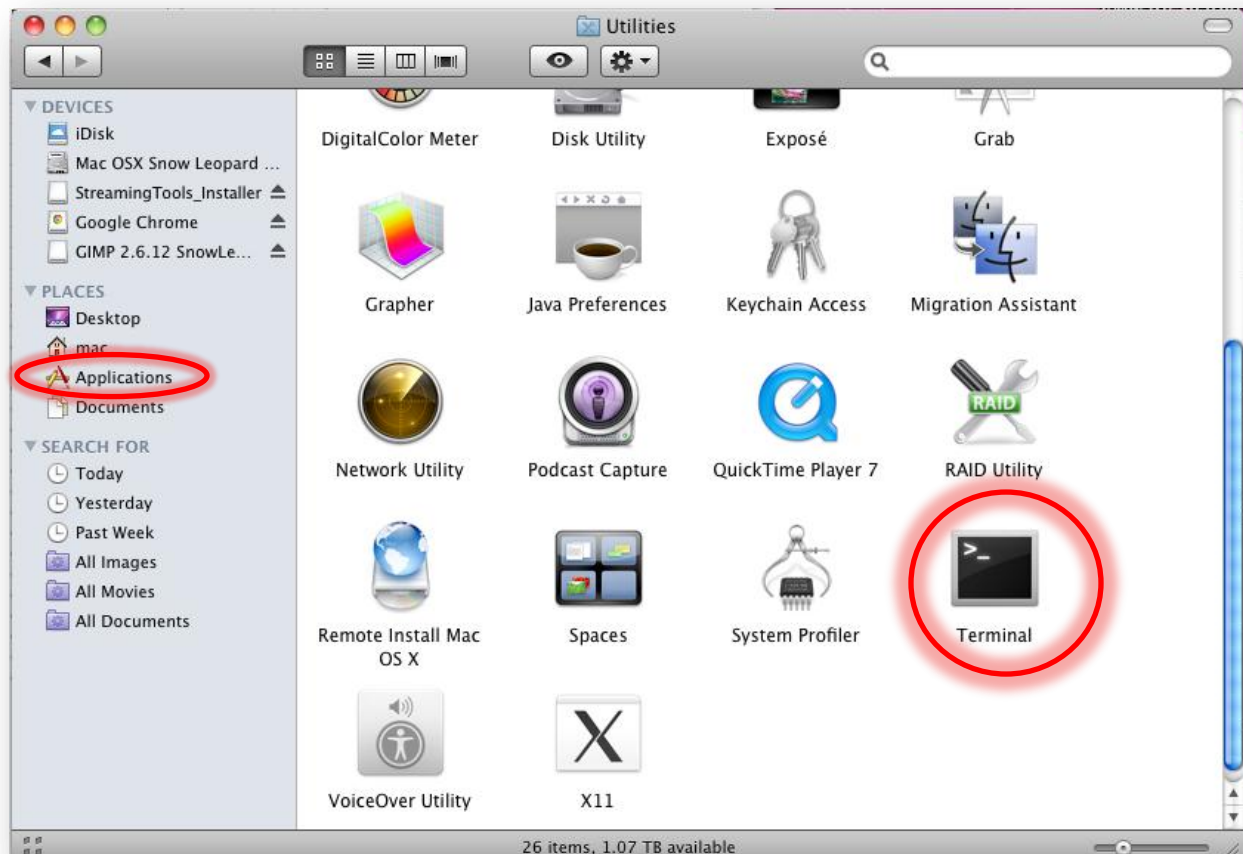
Using HTTP Live Streaming to Convert File

First obtain a video or audio file to convert. These instructions only cover how to convert an audio file, but the process is fairly similar for video files. The specifications for video and audios is as follows:

- Video: H.264 Baseline Level 3.0, Baseline Level 3.1, and Main Level 3.1.
- Audio:
 - HE-AAC or AAC-LC up to 48 kHz, stereo audio
 - MP3 (MPEG-1 Audio Layer 3) 8 kHz to 48 kHz, stereo audio

For audio, this means the file must be an mp3 file or an AAC file. The following demonstration uses an AAC file.

1. Obtain audio file.
2. Place this audio file into a folder you know the location of. To keep it simple, you might want to put it under you “Documents” folder. Remember where you placed this file.
3. Open up a terminal window in Mac by opening “Finder” and going to “Applications” on left side. After this, scroll to “Utilities” and double-click. In this folder, click on “Terminal.”



4. Once the terminal is open, navigate to the location of the audio file. This can be done by typing “cd /path/to/file/”. For example, if you put the audio file under your “Documents” folder as suggested above, the text you would type into the Terminal would be “cd ~/Documents/”. After you type this, press Enter.

“cd” in this case stands for “Change Directory.” It is essentially the same action as clicking on a folder to navigate around your computer. It allows you to enter the location of a folder and navigate to it. Here we navigated to the “Documents” folder. The “~” stands for your home folder. If you have multiple people who use your computer, this ensures you go to your “Documents” folder and not theirs. You can check your current location by typing “pwd” then Enter.

```
macs-Mac:htdocs mac$  
macs-Mac:htdocs mac$ cd ~/Documents/  
macs-Mac:Documents mac$ pwd  
/Users/mac/Documents  
macs-Mac:Documents mac$ █
```

5. Now you need to create a new folder called “stream”. This folder will hold all of the “.ts” files. To make this folder, type “mkdir stream” and Enter. You can confirm that this folder was created by running “ls” which will show you the contents of the current folder.

```
macs-Mac:Documents mac$ ls  
Separate_Ways.aac  
macs-Mac:Documents mac$ mkdir stream  
macs-Mac:Documents mac$ ls  
Separate_Ways.aac      stream  
macs-Mac:Documents mac$ █
```

6. Now you are ready to begin converting the audio file. In this case, This example will be converting the file “Separate_Ways.aac”. Enter the following text, making sure it is all on one line, and press Enter, replacing “FILE_NAME” with the name of the file, in this case “Separate_Ways.aac”. Also, replace “URL” with the url of the website you will be hosting the streaming service from. Make sure the url points to the folder where the “.ts” files will be stored. In the following example, the host website I host the streaming at is “http://fun.rkania.com/hypertext/” and the folder storing the “.ts” files under is “stream”. Therefore the example url is “http://fun.rkania.com/hypertext/stream/”.

```
mediafilesegmenter --audio-only -b URL -I -f ./stream  
FILE_NAME
```

```
Terminal — bash — 84x44
macs-Mac:Documents mac$ mediafilesegmenter --audio-only -b http://fun.rkania.com/hyp
ertext/stream -I -f ./stream Separate_Ways.aac
Apr 21 2012 20:15:36.228: Using floating point is not backward compatible to iOS 4.1
or earlier devices
Apr 21 2012 20:15:36.231: Processing file /Users/mac/Documents/Separate_Ways.aac
Apr 21 2012 20:15:37.200: Finalized /Users/mac/Documents/stream/fileSequence0.aac
Apr 21 2012 20:15:37.347: segment bitrate 253727 is new max
Apr 21 2012 20:15:37.377: Finalized /Users/mac/Documents/stream/fileSequence1.aac
Apr 21 2012 20:15:37.527: segment bitrate 266928 is new max
Apr 21 2012 20:15:37.636: Finalized /Users/mac/Documents/stream/fileSequence2.aac
Apr 21 2012 20:15:37.902: segment bitrate 273968 is new max
Apr 21 2012 20:15:38.194: Finalized /Users/mac/Documents/stream/fileSequence3.aac
Apr 21 2012 20:15:38.483: Finalized /Users/mac/Documents/stream/fileSequence4.aac
Apr 21 2012 20:15:38.679: Finalized /Users/mac/Documents/stream/fileSequence5.aac
Apr 21 2012 20:15:38.685: segment bitrate 299392 is new max
Apr 21 2012 20:15:38.693: Finalized /Users/mac/Documents/stream/fileSequence6.aac
Apr 21 2012 20:15:38.778: Finalized /Users/mac/Documents/stream/fileSequence7.aac
Apr 21 2012 20:15:38.897: Finalized /Users/mac/Documents/stream/fileSequence8.aac
Apr 21 2012 20:15:38.911: Finalized /Users/mac/Documents/stream/fileSequence9.aac
Apr 21 2012 20:15:38.924: Finalized /Users/mac/Documents/stream/fileSequence10.aac
Apr 21 2012 20:15:38.983: Finalized /Users/mac/Documents/stream/fileSequence11.aac
Apr 21 2012 20:15:38.997: Finalized /Users/mac/Documents/stream/fileSequence12.aac
Apr 21 2012 20:15:39.015: Finalized /Users/mac/Documents/stream/fileSequence13.aac
Apr 21 2012 20:15:39.045: Finalized /Users/mac/Documents/stream/fileSequence14.aac
Apr 21 2012 20:15:39.152: Finalized /Users/mac/Documents/stream/fileSequence15.aac
Apr 21 2012 20:15:39.170: Finalized /Users/mac/Documents/stream/fileSequence16.aac
Apr 21 2012 20:15:39.239: Finalized /Users/mac/Documents/stream/fileSequence17.aac
Apr 21 2012 20:15:39.368: Finalized /Users/mac/Documents/stream/fileSequence18.aac
Apr 21 2012 20:15:39.386: Finalized /Users/mac/Documents/stream/fileSequence19.aac
Apr 21 2012 20:15:39.455: Finalized /Users/mac/Documents/stream/fileSequence20.aac
Apr 21 2012 20:15:39.474: Finalized /Users/mac/Documents/stream/fileSequence21.aac
Apr 21 2012 20:15:39.489: Finalized /Users/mac/Documents/stream/fileSequence22.aac
Apr 21 2012 20:15:39.514: Finalized /Users/mac/Documents/stream/fileSequence23.aac
Apr 21 2012 20:15:39.524: Finalized /Users/mac/Documents/stream/fileSequence24.aac
Apr 21 2012 20:15:39.535: Finalized /Users/mac/Documents/stream/fileSequence25.aac
Apr 21 2012 20:15:39.546: Finalized /Users/mac/Documents/stream/fileSequence26.aac
Apr 21 2012 20:15:39.638: Finalized /Users/mac/Documents/stream/fileSequence27.aac
Apr 21 2012 20:15:39.654: Finalized /Users/mac/Documents/stream/fileSequence28.aac
Apr 21 2012 20:15:39.739: Finalized /Users/mac/Documents/stream/fileSequence29.aac
Apr 21 2012 20:15:39.876: Finalized /Users/mac/Documents/stream/fileSequence30.aac
Apr 21 2012 20:15:40.087: Finalized /Users/mac/Documents/stream/fileSequence31.aac
Apr 21 2012 20:15:40.174: Finalized /Users/mac/Documents/stream/fileSequence32.aac
Apr 21 2012 20:15:40.182: average bit rate is 269308 - max file bit rate is 299392
macs-Mac:Documents mac$
```

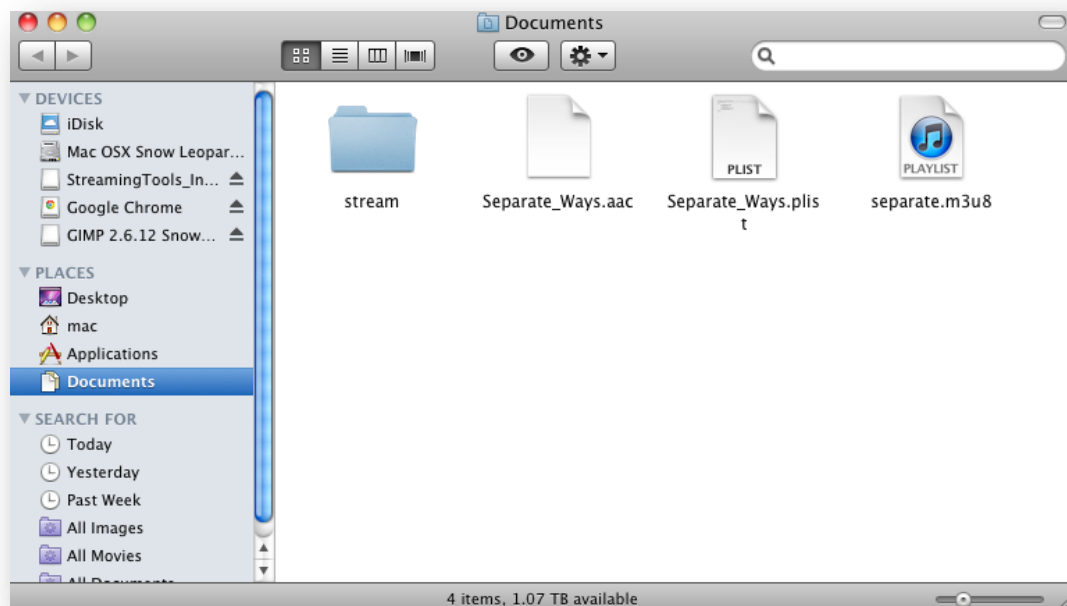
7. This command created a list of files in the folder “stream”. In this case, since the file was an “aac” file, the list of files is also “aac”. If you were converting a video, they would be “.ts” files. It also created an index file that ends in “.m3u8” which is a list of the smaller files in the order to be played.

- Next type the following and press Enter again, replacing URL with the url used above (<http://fun.rkania.com/hypertext/stream/>, making sure to include the “/” at the end). Replace NEW_NAME with the name you want the playlist to have and OLD_NAME with the audio file name excluding the “.aac” at the end.

```
variantplaylistcreator -o NEW_NAME.m3u8  
URL/prog_index.m3u8 ./OLD_NAME.plist
```

```
Apr 19 2012 21:07:36.636: Finalized /Applications/MAMP/htdocs/stream/fileSequence64.aac  
Apr 19 2012 21:07:36.719: Finalized /Applications/MAMP/htdocs/stream/fileSequence65.aac  
Apr 19 2012 21:07:36.725: average bit rate is 269308 - max file bit rate is 299392  
macs-Mac:htdocs mac$  
macs-Mac:htdocs mac$  
macs-Mac:htdocs mac$  
macs-Mac:htdocs mac$  
macs-Mac:htdocs mac$  
macs-Mac:htdocs mac$  
macs-Mac:htdocs mac$  
macs-Mac:htdocs mac$  
macs-Mac:htdocs mac$ variantplaylistcreator -o separate.m3u8 http://fun.rkania.com/hyperte  
xt/stream/prog_index.m3u8 ./Separate_Ways.plist  
macs-Mac:htdocs mac$
```

- Now you should have a new “.m3u8” file in the folder you’ve been working in. You should have a similar set of files as seen below.



Setting up the Web Interface

1. Now that you have the files you need for the audio, create a new HTML file within the same folder name whatever you want. This example uses “SeparateWays.html”. You can either open up a text editor and save the file in this folder or enter the command “touch” as shown below.

```
macs-Mac:Documents mac$  
macs-Mac:Documents mac$ ls  
Separate_Ways.aac      separate.m3u8  
Separate_Ways.plist   stream  
macs-Mac:Documents mac$ touch SeparateWays.html  
macs-Mac:Documents mac$ ls  
SeparateWays.html     Separate_Ways.plist   stream  
Separate_Ways.aac     separate.m3u8  
macs-Mac:Documents mac$
```

2. Now open this file in a text editor and enter the following text, replacing “separate.m3u8” with the “.m3u8” file you created with the NEW_NAME above:

```
<html>  
<head>  
  <title>HTTP Live Streaming Example</title>  
</head>  
<body>  
  <video src="separate.m3u8" height="50" width="400" controls autoplay>  
  </video>  
</body>  
</html>
```

3. Save this file.
4. Upload all of the files in this folder to the url you included above. In this example all the files were uploaded to “<http://fun.rkania.com/hypertext/>”. This can be done using FTP or any other way you access your server.

Testing

HTTP Live Streaming from .m3u8 playlist files will only work on **Mac OS X with Safari or iOS devices, such as iPod Touch, iPhone, and iPad.**

- 1. Open a Safari browser on one of the above devices.**
- 2. Enter the url to the “.html” file you created in the previous section.**
- 3. Click the triangle “Play” button to see if the file plays. If everything works correctly, you should be able to hear the music playing.**

Lessons Learned

Timeline/Schedule

February

Summary: Meeting with the client Keith Gilbertson and with Michael Dunston from Music Department; Researched HTML5 streaming, file formats, and audio conversion

February 3: Met with Keith Gilbertson to discuss starting the project

February 14: Met with Keith Gilbertson and Michael Dunston to discuss the specifics of the project.

March

Summary: Changed topic of research to focus on streaming. Researched Apple's HTTP Live Streaming, Kaltura, Linux HTTP Live Streaming variant.

April

Summary: Implemented Apple's HTTP Live Streaming and showcased our work to Keith Gilbertson

Problems

Requirements:

- Secure Streaming (Legal Issues)
 - When streaming copyrighted material, a requirement for the streaming is that the entirety of the file is not accessible to the user. The typical streaming protocol up to date that follows copyright law is known as RTSP (Real Time Streaming Protocol). However, due to current firewall technology, RTSP is becoming more or less out of date due to its implementation of using multiple ports.
- HTML5
 - HTML4 is slowly being phased out to the more current HTML5. This actually poses its own copyright issues due to the addition of audio and video tags because these tags enable users to access the files on your server directly.
- Mobile Device Functionality
 - Due to advances in mobile technology, the music department wanted to make the streaming content accessible to mobile devices as well. Since the music department's hardware of choice is of the Apple variety, our target devices are Apple products. Also, this bars Flash because Flash is too slow and CPU intensive for mobile devices and not available on most devices.

Unexpected Obstacles:

Based on the original project specification we thought that the focus of the project was on converting the audio files in the recital collection. However, after we discovered what was actually expected of us, we set off on the right track.

Solutions

For our final solution we decided to use Apple's HTTP Live Streaming.

Requirements Addressed:

- Secure Streaming (Legal Issues)
 - Like RTSP protocol, HTTP Live Streaming also chops the media into small bits before sending it to users. However, it is different from RTSP in that it uses http protocol to stream rather than its own unique procedure that isn't friendly with firewalls.
- HTML 5
 - HTTP live streaming was developed by Apple for use with HTML5.
- Mobile Device Functionality
 - Because http live streaming was developed by Apple, it is compatible with Apple devices. HTTP Live Streaming is also more efficient than Flash because it requires no extra software.

Possible Improvements to Solution:

While HTTP Live Streaming works with Safari on Mac and Apple mobile devices, it does not work with other web browsers (such as Firefox, Chrome, and Opera) and other operating systems (such as Linux and Windows). So a possible alternative for other operating systems and web browsers would be to use Flash. However, this doesn't cover mobile devices that aren't Apple products.

Acknowledgements

Client

Keith Gilbertson
Email: keith.gilbertson@vt.edu
Phone: (540) 231-904
Address:
University Libraries
Virginia Tech
Blacksburg, VA 24060

Other

Contact info about:

Nathan Hall
Email: nfhall@vt.edu
Number: (540) 231-1751
Address:
University Libraries
Digital Library and Archives
Blacksburg, VA 24062
United States

Michael Dunston
Email: mdunston@vt.edu
Number: (540) 231-9942
Address:
Music Recording Studio
Blacksburg, VA 24061

References

"Deploying Apples HTTP Live Streaming in a GNU Linux Environment." *Blog.kyri0s.org*. Web. 29 Apr. 2012. <<http://blog.kyri0s.org/post/271121944/deploying-apples-http-live-streaming-in-a-gnu-linux>>.

"HTTP Live Streaming." *HTTP Live Streaming*. Apple Developer. Web. 28 Apr. 2012. <<https://developer.apple.com/resources/http-streaming/>>.

Schulzrinne, Henning. "Real Time Streaming Protocol." *Internet Engineering Task Force*. Columbia University, 4 May 1998. Web. 28 Apr. 2012. <<http://www.ietf.org/rfc/rfc2326.txt>>.