**Streaming**

IP-516\_AppleTV App

**Version number:** 0.1  
**Main author:** AZ, JB  
**Last revision:** 21.10.2016  
**Status:** in work  
**Filename:** Streaming  
**Client:** Michael Keller, UPC

Authors

|  |  |
| --- | --- |
| **Name** | **Initials** |
| Andrea Zirn | AZ |
|  |  |
|  |  |

Revision list

| **Version** | **Date** | **Comment** | **Author** |
| --- | --- | --- | --- |
| 0.1 | 21.10.2016 | Initial version | AZ |
| 0.2 | 28.10.2016 | Add content | AZ |
|  |  |  |  |

Table of Content

Authors 2

Revision list 2

Table of Content 3

1 Heading 1 3

1.1 Heading 2 3

1.1.1 Heading 3 3

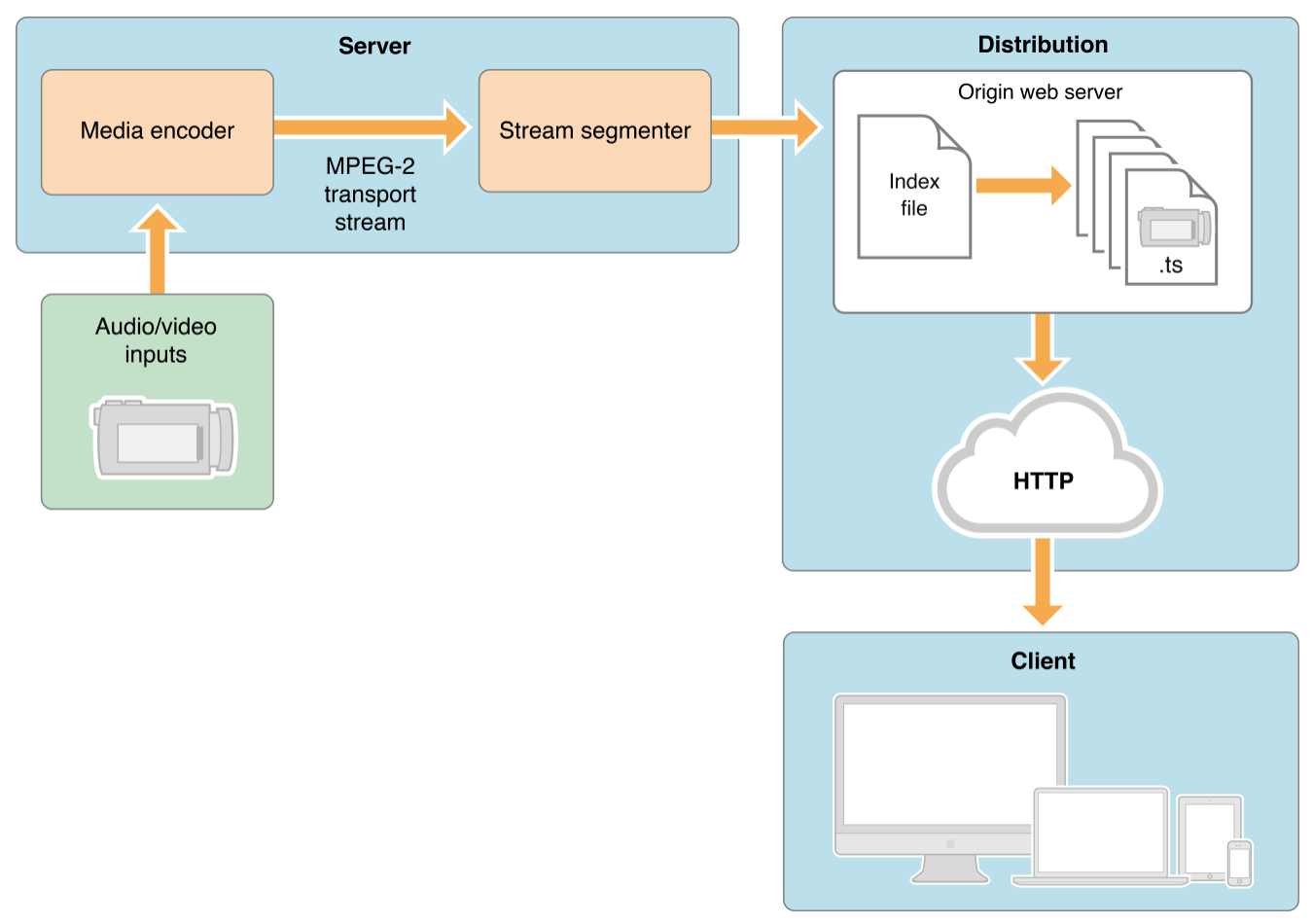
# HTTP Live Streaming[[1]](#footnote-1)

HTTP Live Streaming lets you send audio and video over HTTP from an ordinary web server for playback on iOS-based devices. HTTP Live Streaming supports both live broadcasts and rerecorded content (video on demand).

## General information

* Streaming audio or video to iPhone, iPod, iPad or Apple TV
* Streaming live events without special server software
* Sending video on demand with encryption and authentication
* Apple TV 2 and later includes an HTTP Live Streaming client

### Functionality



You can serve HTTP Live Streaming audio and video from an ordinary web server. HTTP Live Streaming is a way o send audio and video over HTTP from a web server to client software on the desktop or to iOS-based devices.

HTTP Live Streaming sends audio and video as a series of small files, typically of about 10 seconds duration, called media segment files.

### Live Stream

For live streams, Apple provides a free tool to make media segment files and playlists from live MPEG-2 transport streams carrying H.264 video, AAC audio, or MP3 audio. There are a number of hardware and software encoders that can create MPEG-2 transport streams carrying MPEG-4 video and AAC audio in real time.

## HTTP Streaming Architecture[[2]](#footnote-2)

### Overview

HTTP Live Streaming consist of three parts:

1. The server component:  
   is responsible for taking input streams of media and encoding them digitally, encapsulating them in a format suitable for delivery and preparing the encapsulated media for distribution
2. The distribution component:  
   consists of standard web servers. They are responsible for accepting client requests and delivering prepared media and associated resources to the client.
3. The client software:  
   is responsible for determining the appropriate media to request, downloading those resources, and then reassembling them so that the media can be presented to the user in a continuous stream.

Input can be live or from a prerecorded source. It is typically encoded as MPEG-4 (H.264 video and AAC audio) and packaged in an MPEG-2 Transport Stream by off-the-shelf hardware. The MPEG-2 transport stream is then broken into segments and saved as a series of one or more .ts media files. This is typically accomplished using a software tool such as the Apple stream segmenter.

### Server Components

The server requires a media encoder, which can be off-the-shelf hardware, and a way to break the encoded media into segments and save them as files, which can either be software such as the media stream segmenter provided by Apple or part of an integrated third-party solution.

**Media Encoder**

The media encoder takes a real-time signal from an audio-video device, encodes the media, and encapsulates it for transport. Currently, the supported delivery format is MPEG-2 Transport Streams for audio-video.

The encoder delivers the encoded media in an MPEG-2 Transport Stream over the local network to the stream segmenter. The transport stream is a packaging format that can be used with a number of different compression formats. For video: The UIKit UIIMagePickerController, AVKit, AV Foundation, Core Media. (for more information see: <https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/MediaLayer/MediaLayer.html#//apple_ref/doc/uid/TP40007898-CH9-SW6>)

**Stream Segmenter**

The stream segmenter is a process that reads the Transport Stream from the local network and divides it into a series of small media files of equal duration. The segmenter also creates an index file containing references to the individual media files. Each time the segmenter completes a new media file, the index file is updated. The index is used to track the availability and location of the media files. Media segments are saved as .ts-files and index files are saved as .M3U8-playlists.

**File Segmenter**

If you already have a media file encoded using supported codecs, you can use a file segmenter to encapsulate it in an MPEG-2 transport stream and break it into segments of equal length. The file segmenter allows you to use a library of existing audio and video files for sending video on demand via HTTP Live Streaming. The file segmenter performs the same tasks as the stream segmenter, but it takes files as input instead of streams.

### Distribution Components

The distribution system is a web server or a web caching system that delivers the media files and index files to the client over HTTP.

### Client Components

The client software begins by fetching the index file, based on a URL identifying the stream. The index file in turn specifies the location of the available media files, decryption keys, and any alternate streams available. For the selected stream, the client downloads each available media file in sequence. Each file contains a consecutive segment of the stream. Once it has a sufficient amount of data downloaded, the client begins presenting the reassembled stream to the user.

The client is responsible for fetching any decryption keys, authenticating or presenting a user interface to allow authentication, and decrypting media files as needed.

## Using HTTP Live Stream[[3]](#footnote-3)

The HTTP Live Streaming Tools package installs prerelease command-line tools that are used for deployment and validation of HTTP Live Streaming solutions. The tools are:

* **Media Stream Segmenter**  
  It takes an MPEG-2 transport stream as an input and produces a series of equal-length files from it, suitable for use in HTTP Live Streaming. It can also generate index files (also known as playlists), encrypt the media, produce encryption keys, optimize the files by reducing overhead, and create the necessary files for automatically generating multiple stream alternates.
* **Media File Segmenter**  
  It takes an encoded media file as an input, wraps it in an MPEG-2 transport stream, and produces a series of equal-length files from it, suitable for use in HTTP Live Streaming. The media file segmenter can also produce index files (playlists) and decryption keys. The file segmenter behaves very much like the stream segmenter, but it works on existing files instead of streams coming from an encoder.
* **Media Subtitle Segmenter**
* **Variant Playlist Creator**  
  It creates a master index file, or playlist, listing the index files for alternate streams at different bit rates, using the output of the Media File Segmenter.
* **Media Stream Validator**  
  It examines the index files, stream alternates, and media segment files on a server and tests to determine whether they will work with HTTP Live Streaming clients.
* HLS Report
* ID3 Tag Generator (or a variant playlist generator)  
  generates ID3 metadata tags. These tags can either be written to a file or inserted into outgoing stream segments

This tools can be downloaded from the Apple Developer website.[[4]](#footnote-4)

**1.3.1 Session Types**

The HTTP Live Streaming protocol supports two types of sessions: events and video on demand.

**VOD sessions**

For VOD sessions, media files are available representing the entire duration of the presentation. The index file is static and contains a complete list of all files created since the beginning of the presentation. HTTP Live Streaming offers advantages over progressive download for VOD, such as support for media encryption and dynamic switching between streams of different data rates in response to changing connection speeds

**Live sessions**

Live sessions can be presented as a complete record of an event, or as a sliding window with a limited time range the user can seek within. For live sessions, as new media files are created and made available, the index file is updated. The new index file lists the new media files. Older media files can be removed from the index and discarded. Alternatively, the index can simply add new media files to the existing list—this type of session can be easily converted to VOD after the event completes.

**Broadcast to VOD**

To convert a live broadcast to VOD, do not remove the old media files from the server or delete their URLs from the index file; instead, add an #EXT-X-ENDLIST tag to the index when the event ends. This allows clients to join the broadcast late and still see the entire event. It also allows an event to be archived for rebroadcast with no additional time or effort.

If your playlist contains an EXT-X-PLAYLIST-TYPE tag, you should also change the value from EVENT to VOD.

**1.3.2 Content Protection**

Media files containing stream segments may be individually encrypted. When encryption is employed, references to the corresponding key files appear in the index file so that the client can retrieve the keys for decryption. Currently HTTP Live Streaming supports AES-128 encryption using 16-octet keys. The format of the key file is a packed array o

f these 16 octets in binary format. The media stream segmenter provides encryption and supports three modes for configuring encryption.

1. Mode that allows to specify a path to an existing key file on disk. In this mode the segmenter inserts the URL of the existing key file in the index file. It encrypts all media files using this key.
2. Mode that instructs the segmenter to generate a random key file, save this in a specified location and reference it in the index file.
3. Mode that instructs the segmenter to generate a new random key file every n media segments, save it in a specified location, and reference it in the index file. This mode is referred to as key rotation. Each group of n files is encrypted using a different key.

**1.3.3 Caching and Delivery Protocols**

it is important to make sure that any content delivery network you use understands that the .M3U8 index files are not to be cached for longer than one media segment duration for live broadcasts, where the index file is changing dynamically.

**1.3.4 Stream Alternates**

A master index file may reference alternate streams of content. References can be used to support delivery of multiple streams of the same content with varying quality levels for different bandwidths or devices. HTTP Live Streaming supports switching between streams dynamically if the available bandwidth changes.

The client software uses heuristics to determine appropriate times to switch between the alternates. Currently, these heuristics are based on recent trends in measured network throughput.

The master index file points to alternate streams of media, both are in .M3U8 playlist format. The master file is downloaded once, but for live broadcasts the alternate index are reloaded periodically. The first alternate listed in the master is the first stream used, after that, the client chooses among the alternates by available bandwidth.

Creating set of stream alternates:

* use the variantplaylistcreator tool
* specify the –generate-variant-playlist option for either the mediafilesegmenter tool or the mediastreamsegmenter tool

Best practice:

* when possible, encode enough variants to provide the best quality stream across a wide range of connection speeds (150, 350, 550, 900, 1500 kbps)
* use relative path names in variant playlists and in the individual .M3U8 playlist files
* the video aspect ratio on alternate streams must be exactly the same
* the RESOLUTION field in the EXT-X-STREAM\_INF should be included to help the client choose an appropriate stream
* more than one master index file consisting of the same alternate index files but with a different first stream is recommended:   
  150k stream for the cellular variant playlist, 240k/440k for the Wi-Fi variant playlist

**1.3.5 Requirements for Apps**

The following requirements apply to iOS apps submitted for distribution in the App Store for use on Apple products:

* You are required to use HTTP Live Streaming if your app delivers video over cellular network, and the video exceed either 10 minutes duration or 5 MB of data in a five minute period.
* If your app uses HTTP Live Streaming over cellular networks, you are also required to provide at least on stream at 64 Kbps or lower bandwidth.

**1.3.6 Redundant Streams**

If the playlist contains alternate streams, they can not only operate as bandwidth or device alternates, but also as failure fallbacks. In case where the client is unable to reload the index file for the stream, the client attempts to switch to an alternate stream.

In the event of an index load failure on one stream, the client chooses the highest bandwidth alternate stream that the network connection supports. If there are multiple alternates at the same bandwidth, the client chooses among them in the order listed in the playlist.

How to support redundant streams?

* Create a stream, or multiple alternate bandwidth streams and generate a playlist file
* Create parallel stream, or s set of streams, on a separate server or content distribution service
* Add the list of backup streams of the playlist file, so that the backup stream at each bandwidth is listed after the primary stream (more than one is possible as well)

Sample code: (ALPHA is the primary stream and BETA is the backup stream)

|  |
| --- |
| #EXTM3U |
| #EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=200000, RESOLUTION=720x480 |
| http://ALPHA.mycompany.com/lo/prog\_index.m3u8 |
| #EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=200000, RESOLUTION=720x480 |
| http://BETA.mycompany.com/lo/prog\_index.m3u8 |
|  |
| #EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=500000, RESOLUTION=1920x1080 |
| http://ALPHA.mycompany.com/md/prog\_index.m3u8 |
| #EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=500000, RESOLUTION=1920x1080 |
| http://BETA.mycompany.com/md/prog\_index.m3u8 |

**1.3.7 Adding Timed Metadata**

Various kinds of metadata can be added to media stream segments. Wirting a own client software, however, using either MPMoviePlayerController or AVPlayerItem, streamed metadata can be accessed using the timedMetaData property.

When using Apple tools there is the possibility to add timed metadata (I is inserted into a media stream at a given time offset) by specifying a metadata file to either the stream segmenter or the file segmenter. Metadata specified this way is automatically inserted into every media segment.

2 variants of Adding timed metadata:

1. using the id3taggenerator tool, with its output set to the stream segmenter. The tool generates ID3 metadata and passes it the stream segmenter for inclusion in the outbound stream.
2. Using the file segmenter. See: https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html#//apple\_ref/doc/uid/TP40008332-CH102-SW1

Once metadata has been inserted into a media segment, it is persistent. If a live broadcast is re-purposed as video on demand, for example, it retains any metadata inserted during the original broadcast.

**1.3.8 Adding Closed Captions**

HTTP Live Streaming supports closed captions within streams.

If you are using the stream segmenter, you need to add CEA-608 closed captions to the MPEG-2 transport stream (in the main video elementary stream) as specified in ATSC A/72.

If you are using the file segmenter, you should encapsulate your media in a QuickTime movie file and add a closed captions track ('clcp').

If you are writing an app, the AVFoundation framework supports playback of closed captions.

Live streaming also supports multiple subtitle and closed caption tracks in Web Video Text Tracks (WebVTT) format.

Sample code: master playlist with multiple closed caption tracks:

|  |
| --- |
| #EXTM3U |
|  |
| #EXT-X-MEDIA:TYPE=CLOSED-CAPTIONS,GROUP-ID="cc",NAME="CC1",LANGUAGE="en",DEFAULT=YES,AUTOSELECT=YES,INSTREAM-ID="CC1" |
| #EXT-X-MEDIA:TYPE=CLOSED-CAPTIONS,GROUP-ID="cc",NAME="CC2",LANGUAGE="sp",AUTOSELECT=YES,INSTREAM-ID="CC2" |
|  |
| #EXT-X-STREAM-INF:BANDWIDTH=1000000,SUBTITLES="subs",CLOSED-CAPTIONS="cc" |
| x.m3u8 |

In the encoding process, the WebVTT files are broken into segments just as audio and video media. The resulting media playlist includes segment durations to sync text with the correct point in the associated video.

**1.3.9 Preparing Media for Delivery to iOS-Based Devices**

The recommended encoder settings for streams used with iOS-based devices are shown in the following four tables. For live streams, these settings should be available from your hardware or software encoder. If you are re-encoding from a master file for video on demand, you can use a video editing tool such as Compressor.

File format for the file segmenter can be a QuickTime movie, MPEG-4 video, or MP3 audio, using the specified encoding.

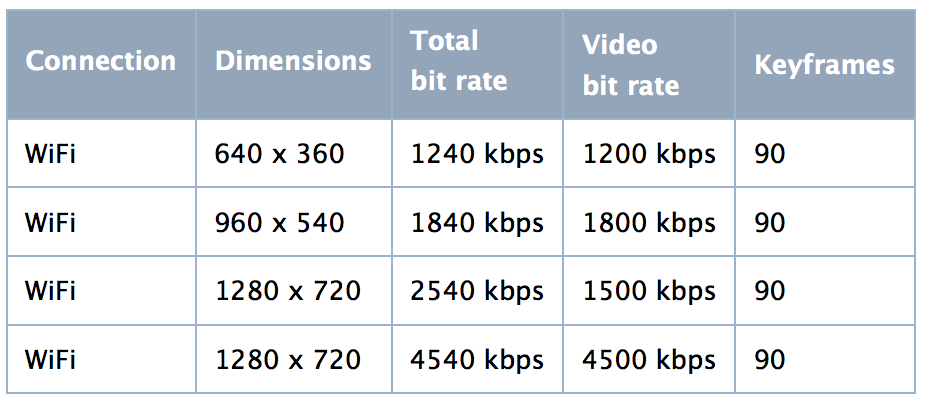
Stream format for the stream segmenter must be MPEG elementary audio and video streams, wrapped in an MPEG-2 transport stream, and using the following encoding (focused on Apple TV)

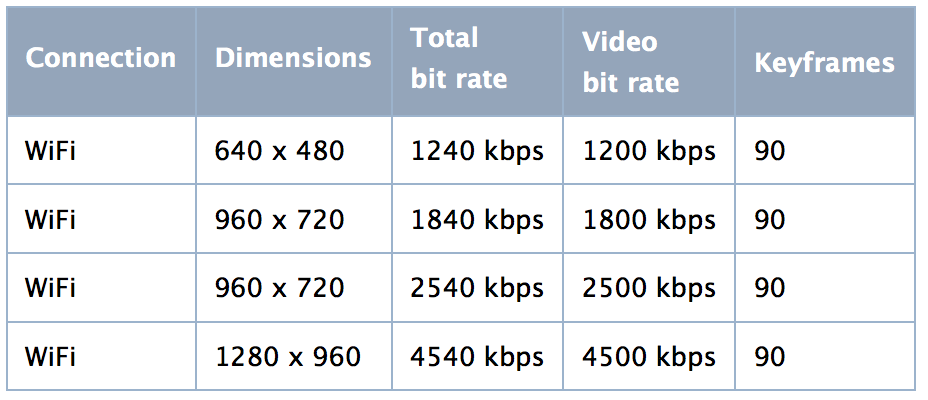
* H.264 Main profile 3.1: Apple TV 2 and later
* H.264 Main Profile 4.0: Apple TV 3 and later
* H.264 High Profile 4.0: Apple TV 3 and later

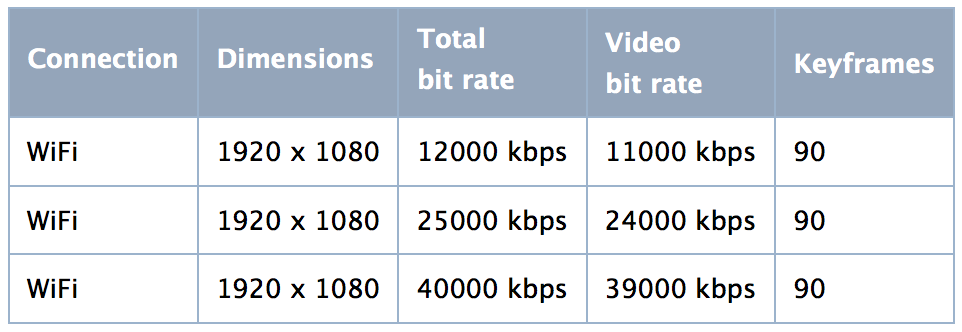
A frame rate of 10 fps is recommended for video streams under 200 kbps. For video streams under 300 kbps, a frame rate of 12 to 15 fps is recommended. For all other streams, a frame rate of 29.97 is recommended.

Encode audio as either of the following:

* HE-AAC or AAC-LC, stereo
* MP3 (MPEG-1 Audio Layer 3), stereo

Additional main profile encoder settings, 16:9 aspect ratio

Additional main profile encoder settings, 4:3 aspect ratio

Additional high profile encoder settings, 16:9 aspect ratio

**1.3.10 Sample Streams**

There are a series of HTTP streams available for testing on Apple’s developer site.

https://developer.apple.com/streaming/

## Deploying HTTP Live Streaming

To actually deploy HTTP Live Streaming, you need to create either an HTML page for browsers or a client app to act as a receiver. You also need the use of a web server and a way to either encode live streams as MPEG-2 transport streams or to create MP3 or MPEG-4 media files with H.264 and AAC encoding from your source material.

1. <https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/Introduction/Introduction.html> (21.10.2016) [↑](#footnote-ref-1)
2. <https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/HTTPStreamingArchitecture/HTTPStreamingArchitecture.html#//apple_ref/doc/uid/TP40008332-CH101-SW2> (21.10.2016) [↑](#footnote-ref-2)
3. <https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html#//apple_ref/doc/uid/TP40008332-CH102-SW1> (28.10.2016) [↑](#footnote-ref-3)
4. <https://developer.apple.com/programs/how-it-works/> (28.10.2016) [↑](#footnote-ref-4)