Informational
Internet-Draft
Intended status: Informational
Expires: November 2, 2009

R. Pantos, Ed.
Apple Inc.

May 1, 2009

HTTP Live Streaming
draft-pantos-http-live-streaming-00

Status of this Memo

Copyright Notice

Abstract

This document describes a protocol for transmitting unbounded streams of multimedia data over HTTP. It specifies the data format of the files and the actions to be taken by the server (sender) and the clients (receivers) of the streams. It describes version 1.0 of this protocol.

Table of Contents

1. Introduction

This document describes a protocol for transmitting unbounded streams of multimedia data over HTTP [RFC2616]. The protocol supports the encryption of media data, and the provision of alternate versions (e.g. bitrates) of a stream. Media data can be transmitted soon after it is created, allowing it to be received in near real-time.

External references that describe related standards such as HTTP are listed in Section 11.

2. Summary

A multimedia presentation is specified by a URI [RFC3986] to a Playlist file, which is an ordered list of additional URIs. Each URI in the Playlist file refers to a media file which is a segment of a single contiguous stream.

To play the stream, the client first obtains the Playlist file and then obtains and plays each media file in the Playlist. It reloads the Playlist file as described in this document to discover additional segments.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. The Playlist file

Playlists MUST be Extended M3U Playlist files [M3U]. This document extends the M3U file format by defining additional tags.

An M3U Playlist is a text file that consists of individual lines. Lines are terminated by either a single LF character or a CR character followed by an LF character. Each line is a URI, a blank, or starts with the comment character '#'. URIs identify media files to be played. Blank lines are ignored.

Lines that start with the comment character '#' are either comments or tags. Tags begin with #EXT. All other lines that begin with '#' are comments and SHOULD be ignored.

Implementations SHOULD produce Playlist files encoded in UTF-8 [RFC3629]. URIs to such Playlist files SHOULD end in .m3u8 and/or have the MIME type [RFC2046] application/x-mpegURL.

The Extended M3U file format defines two tags: EXTM3U and EXTINF. An Extended M3U file is distinguished from a basic M3U file by its first line, which MUST be #EXTM3U.

EXTINF is a record marker that describes the media file identified by the URI that follows it. Each media file URI MUST be preceded by an EXTINF tag. Its format is:

#EXTINF:<duration>,<title>

"duration" is an integer that specifies the duration of the media file in seconds. Durations SHOULD be rounded to the nearest integer. The remainder of the line following the comma is the title of the media file.

3.1. New Tags

This document defines seven new tags: EXT-X-TARGETDURATION, EXT-X-MEDIA-SEQUENCE, EXT-X-KEY, EXT-X-PROGRAM-DATE-TIME, EXT-X-ALLOW-CACHE, EXT-X-STREAM-INF, and EXT-X-ENDLIST.

3.1.1. EXT-X-TARGETDURATION

The EXT-X-TARGETDURATION tag indicates the approximate duration of the next media file that will be added to the main presentation. It MUST appear in the Playlist file. Its format is:

#EXT-X-TARGETDURATION:<seconds>

The actual duration of the media file MAY differ slightly from the target duration.

3.1.2. EXT-X-MEDIA-SEQUENCE

Each media file URI in a Playlist has a unique sequence number. The sequence number of a URI is equal to the sequence number of the URI that preceded it plus one. The EXT-X-MEDIA-SEQUENCE tag indicates the sequence number of the first URI that appears in a Playlist file. Its format is:

#EXT-X-MEDIA-SEQUENCE:<number>

If the Playlist file does not contain an EXT-X-MEDIA-SEQUENCE tag then the sequence number of the first URI in the playlist SHALL be considered to be 1.

See Section 6.2.1 and Section 6.2.4 for information on handling the EXT-X-MEDIA-SEQUENCE tag.

3.1.3. EXT-X-KEY

Media files MAY be encrypted. The EXT-X-KEY tag provides information necessary to decrypt media files that follow it. Its format is:

#EXT-X-KEY:METHOD=<method>[,URI="<URI>"]

The METHOD parameter specifies the encryption method. The URI parameter, if present, specifies how to obtain the key.

Version 1.0 of the protocol defines two encryption methods: NONE and AES-128. An encryption method of NONE means that media files are not encrypted. An encryption method of AES-128 means that media files are encrypted using the Advanced Encryption Standard [AES_128] with a 128-bit key.

A new EXT-X-KEY supersedes any prior EXT-X-KEY.

If no EXT-X-KEY tag is present then media files are not encrypted.

See Section 5 for the format of the key file, and Section 5.1, Section 6.1.2 and Section 6.2.5 for additional information on media file encryption.

3.1.4. EXT-X-PROGRAM-DATE-TIME

The EXT-X-PROGRAM-DATE-TIME tag associates the beginning of the next media file with an absolute date and/or time. The date/time representation is ISO/IEC 8601:2004 [ISO_8601] and SHOULD indicate a time zone. For example:

#EXT-X-PROGRAM-DATE-TIME:<YYYY-MM-DDThh:mm:ssZ>

3.1.5. EXT-X-ALLOW-CACHE

The EXT-X-ALLOW-CACHE tag indicates whether the client MAY cache downloaded media files for later replay. Its format is:

#EXT-X-ALLOW-CACHE:<YES|NO>

3.1.6. EXT-X-ENDLIST

The EXT-X-ENDLIST tag indicates that no more media files will be added to the Playlist file. Its format is:

#EXT-X-ENDLIST

3.1.7. EXT-X-STREAM-INF

The EXT-X-STREAM-INF tag indicates that the next URI in the Playlist file identifies another Playlist file. Its format is:

#EXT-X-STREAM-INF:[attribute=value][,attribute=value]*
<URI>

The following attributes are defined for the EXT-X-STREAM-INF tag:

BANDWIDTH=<n>

where n is an approximate upper bound of the stream bitrate, expressed as a number of bits per second.

PROGRAM-ID=<i>

where i is a number that uniquely identifies a particular presentation within the scope of the Playlist file.

A Playlist file MAY contain multiple EXT-X-STREAM-INF URIs with the same PROGRAM-ID to describe variant streams of the same presentation.

CODECS="[format][,format]*"

where each format specifies a media sample type that is present in a media file in the Playlist file.

Valid format identifiers are those in the ISO File Format Name Space defined by RFC 4281 [RFC4281].

4. Media files

Each media file URI in a Playlist file MUST identify a media file which is a segment of the overall presentation. Each media file MUST be formatted as an MPEG-2 Transport Stream, an MPEG-2 Program Stream, or an MPEG-2 audio elementary stream [ISO_13818]. All media files in a presentation MUST have the same format.

Transport Stream files MUST contain a single MPEG-2 Program. Clients SHOULD be prepared to handle multiple tracks of a particular type (e.g. audio or video) by choosing a reasonable subset. Clients SHOULD ignore private streams that they do not recognize inside Transport Streams.

The encoding parameters for samples within a stream inside a media file and between corresponding streams across multiple media files

SHOULD remain consistent. However clients SHOULD deal with encoding changes as they are encountered, for example by scaling video content to accomodate a resolution change.

5. Key files

An EXT-X-KEY tag with the URI parameter identifies a Key file. A Key file contains the cipher key that MUST be used to decrypt subsequent media files in the Playlist.

The AES-128 encryption method uses 16-octet keys. The format of the Key file is simply a packed array of these 16 octets in binary format.

5.1. IV for AES-128

128-bit AES requires the same 16-octet Initialization Vector (IV) to be supplied when encrypting and decrypting. Varying this IV increases the strength of the cipher.

When using the encryption METHOD AES-128, implementations SHALL use the sequence number of the media file as the IV when encrypting or decrypting media files. The big-endian binary representation of the sequence number SHALL be placed in a 16-octet buffer and padded (on the left) with zeros.

6. Client/Server Actions

This section describes how the server generates the Playlist and media files and how the client should download and play them.

6.1. Server Process

The production of the MPEG-2 stream is outside the scope of this document, which simply presumes a source of a continuous stream containing the main presentation.

The server MUST divide the stream into individual media files whose duration is approximately equal. The server SHOULD attempt to divide the stream at points that support effective decode of individual media files, e.g. on packet and key frame boundaries.

The server MUST create a URI for each media file that will allow its clients to obtain the file.

The server MUST create a Playlist file. The Playlist file MUST

conform to the format described in Section 3. A URI for each media file that the server wishes to make available MUST appear in the Playlist in the order in which it is to be played. The entire media file MUST be available to clients if its URI is in the Playlist file.

The Playlist file MUST contain an EXT-X-TARGETDURATION tag. It MUST indicate the approximate duration of the next media file to be added to the main presentation. This value MUST remain constant for the entire presentation. A typical target duration is 10 seconds.

The server MUST create a URI for the Playlist file that will allow its clients to obtain the file.

Changes to the Playlist file MUST be made atomically from the point of view of the clients.

Every media file URI in a Playlist MUST be prefixed with an EXTINF tag indicating the approximate duration of the media file.

The server MAY associate an absolute date and time with a media file by prefixing its URI with an EXT-X-PROGRAM-DATE-TIME tag. The value of the date and time is arbitrary.

If the Playlist contains the final media file of the presentation then the Playlist file MUST contain the EXT-X-ENDLIST tag.

If the server wishes to remove an entire presentation, it MUST make the Playlist file unavailable to clients. It SHOULD ensure that all media files in the Playlist file remain available to clients for at least the duration of the Playlist file at the time of removal.

6.1.1. Sliding Window Playlists

The server MAY limit the availability of media files to those which have been most recently added to the Playlist. To do so the Playlist file MUST ALWAYS contain exactly one EXT-X-MEDIA-SEQUENCE tag. Its value MUST be incremented by 1 for every media file URI that is removed from the Playlist file.

Media file URIs MUST be removed from the Playlist file in the order in which they were added.

When the server removes a media file URI from the Playlist, the media file MUST remain available to clients for a period of time equal to the duration of the media file plus the duration of the longest Playlist file in which the media file has appeared. The duration of a Playlist file is the sum of the durations of the media files within it.

If a server plans to remove a media file, it SHOULD ensure that an HTTP Expires header reflects the planned time-to-live when it is delivered to clients.

The server MUST maintain at least three main presentation media files in the Playlist at all times unless the EXT-X-ENDLIST tag is present.

6.1.2. Encrypting media files

If media files are to be encrypted the server MUST define a URI which will allow authorized clients to obtain a Key file containing a decryption key. The Key file MUST conform to the format described in Section 5.

The server MAY set the Expires header in the key response to indicate that the key may be cached.

If the encryption METHOD is AES-128, AES-128 CBC encyption SHALL be applied to individual media files. The entire file MUST be encrypted. Cipher Block Chaining MUST NOT be applied across media files. The sequence number of the media file MUST be used as the IV as described in Section 5.1.

The server MUST add an EXT-X-KEY tag with the key URI to the end of the Playlist file. The server MUST encrypt all subsequent media files with that key until a change in encryption configuration is desired.

If the server wishes to switch to a new encryption key it MUST make the new key available via a new URI which is distinct from all previous key URIs used by that presentation. It MUST add an EXT-X-KEY tag with the new key URI to the end of the Playlist file. It MUST encrypt all subsequent media files with that key until a change in encryption configuration is desired.

If the server wishes to turn off encryption it MUST add an EXT-X-KEY tag with the encryption METHOD NONE to the end of the Playlist file. It MUST NOT contain a URI parameter. All subsequent media files MUST be cleartext (not encrypted) until a change in encryption configuration is desired.

The server MUST NOT remove an EXT-X-KEY tag from the Playlist file if the Playlist file contains a URI to a media file encrypted with that key.

6.1.3. Providing variant streams

A server MAY offer multiple Playlist files to provide different encodings of the same presentation. If it does so it SHOULD provide a variant Playlist file that lists each variant stream to allow clients to switch between encodings dynamically.

Variant Playlists MUST contain an EXT-X-STREAM-INF tag for each variant stream. Each EXT-X-STREAM-INF tag for the same presentation MUST have the same PROGRAM-ID attribute value. The PROGRAM-ID value for each presentation MUST be unique within the variant Playlist.

If an EXT-X-STREAM-INF tag contains the CODECS attribute, the attribute value MUST include every format defined by [RFC4281] that is present in any media file that appears or will appear in the Playlist file.

The server MUST meet the following constraints when producing variant streams:

- Each variant stream MUST consist of the same content, including content which is not part of the main presentation.

- The server MUST make the same period of content available for all variant streams, within an accuracy of the smallest target duration of the streams.

- The media files of variant streams MUST be either MPEG-2 Transport Streams or MPEG-2 Program Streams. Their sample timestamps MUST match the timestamps of corresponding content in all other variant streams.

In addition, all variant streams SHOULD contain the same audio encoding. This allows clients to switch between streams without audible glitching.

6.2. Client Process

How the client obtains the URI to the Playlist file is outside the scope of this document; it is presumed to have done so.

The client MUST obtain the Playlist file from the URI. If the Playlist file so obtained is a variant Playlist, the client MUST obtain the Playlist file from the variant Playlist.

This document does not specify the treatment of variant streams by clients.

6.2.1. Loading the Playlist file

Every time a Playlist file is loaded or reloaded from the Playlist URI:

The client SHOULD check that the Playlist file begins with #EXTM3U and refuse to continue if it does not. The client SHOULD ignore any tags it does not recognize.

The client MUST determine the next media file to load as described in Section 6.2.4.

If the Playlist contains the EXT-X-MEDIA-SEQUENCE tag, the client SHOULD assume that each media file in it will become unavailable at the time that the Playlist file was loaded plus the duration of the Playlist file. The duration of a Playlist file is the sum of the durations of the media files within it.

6.2.2. Playing the Playlist file

The client SHALL choose which media file to play first from the Playlist when playback starts. If the Playlist file contains the EXT-X-ENDLIST tag, any file in the Playlist MAY be played first. If the EXT-X-ENDLIST tag is not present, any file except for the last and second-to-last files in the Playlist MAY be played first.

Once the first media file to play has been chosen, subsequent media files in the Playlist MUST be loaded in the order that they appear and played in the order that they are loaded.

The client SHOULD attempt to load media files in advance of when they will be required for uninterrupted playback to compensate for temporary variations in latency and throughput.

If the Playlist file contains the EXT-X-ALLOW-CACHE tag and its value is NO, the client MUST NOT cache downloaded media files after they have been played. Otherwise the client MAY cache downloaded media files indefinitely for later replay.

The client MAY use the value of the EXT-X-PROGRAM-DATE-TIME tag to display the program origination time to the user. If the value includes time zone information the client SHALL take it into account, but if it does not the client MUST NOT infer an originating time zone.

The client MUST NOT depend upon the correctness or the consistency of the value of the EXT-X-PROGRAM-DATE-TIME tag.

6.2.3. Reloading the Playlist file

The client MUST periodically reload the Playlist file unless it contains the EXT-X-ENDLIST tag.

However the client MUST NOT attempt to reload the Playlist file more frequently than specified by this section.

When a client loads a Playlist file for the first time or reloads a Playlist file and finds that it has changed since the last time it was loaded, the client MUST wait for a period of time before attempting to reload the Playlist file again. This period is called the initial minimum reload delay. It is measured from the time that the client began loading the Playlist file.

The initial minimum reload delay is the duration of the last media file in the Playlist or 3 times the target duration, whichever is less. Media file duration is specified by the EXTINF tag.

If the client reloads a Playlist file and finds that it has not changed then it MUST wait for a period of time before retrying. The minimum delay is three times the target duration or a multiple of the initial minimum reload delay, whichever is less. This multiple is 0.5 for the first attempt, 1.5 for the second, and 3.0 thereafter.

6.2.4. Determining the next file to load

The client MUST examine the Playlist file every time it is loaded or reloaded to determine the next media file to load.

The first file to load MUST be the file that the client has chosen to play first, as described in Section 6.2.2.

If the first file to be played has been loaded and the Playlist file does not contain the EXT-X-MEDIA-SEQUENCE tag then the client MUST verify that the current Playlist file contains the URI of the last loaded media file at the offset it was originally found at, halting playback if it does not. The next media file to load MUST be the first media file URI following the last-loaded URI in the Playlist.

If the first file to be played has been loaded and the Playlist file contains the EXT-X-MEDIA-SEQUENCE tag then the next media file to load SHALL be the one with the lowest sequence number that is greater than the sequence number of the last media file loaded.

6.2.5. Playing encrypted media files

If a Playlist file contains an EXT-X-KEY tag that specifies a Key file URI, the client MUST obtain that key file and use the key inside it to decrypt all media files following the EXT-X-KEY tag until another EXT-X-KEY tag is encountered.

If the encryption METHOD is AES-128, AES-128 CBC decryption SHALL be applied to individual media files. The entire file MUST be decrypted. Cipher Block Chaining MUST NOT be applied across media files. The sequence number of the media file MUST be used as the IV as described in Section 5.1.

If the encryption METHOD is NONE, the client MUST treat all media files following the EXT-X-KEY tag as cleartext (not encrypted) until another EXT-X-KEY tag is encountered.

7. Examples

This section contains several example Playlist files.

7.1. Simple Playlist file

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXTINF:5220,
http://media.example.com/entire.ts
#EXT-X-ENDLIST
```

7.2. Sliding Window Playlist, using HTTPS

```
#EXTM3U
#EXT-X-TARGETDURATION:8
#EXT-X-MEDIA-SEQUENCE:2680

#EXTINF:8,
https://priv.example.com/fileSequence2680.ts
#EXTINF:8,
https://priv.example.com/fileSequence2681.ts
#EXTINF:8,
https://priv.example.com/fileSequence2682.ts
```

7.3. Playlist file with encrypted media files

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:7794
#EXT-X-TARGETDURATION:15

#EXT-X-KEY:METHOD=AES-128,URI="https://priv.example.com/key.php?r=52"

#EXTINF:15,
http://media.example.com/fileSequence7794.ts
#EXTINF:15,
http://media.example.com/fileSequence7795.ts
#EXTINF:15,
http://media.example.com/fileSequence7796.ts

#EXT-X-KEY:METHOD=AES-128,URI="https://priv.example.com/key.php?r=53"

#EXTINF:15,
http://media.example.com/fileSequence7797.ts
```

7.4. Variant Playlist file

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1280000
http://example.com/low.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=2560000
http://example.com/mid.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=7680000
http://example.com/hi.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=65000,CODECS="mp4a.40.5"
http://example.com/audio-only.m3u8
```

8. Contributors

Significant contributions to the design of this protocol were made by Jim Batson, David Biderman, Bill May, Roger Pantos, and Alan Tseng.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

Since the protocol relies primarily on HTTP for transport, most of the same security considerations apply. See section 15 of RFC 2616

[RFC2616].

Media file parsers are typically subject to "fuzzing" attacks. Clients should take care when parsing files received from a server so that non-compliant files are rejected.

11. References

11.1. Normative References

[AES_128] U.S. Department of Commerce/National Institute of Standards and Technology, "Advanced Encryption Standard (AES), FIPS PUB 197", November 2001, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[ISO_13818] International Organization for Standardization, "ISO/IEC International Standard 13818; Generic coding of moving pictures and associated audio information", November 1994, <www.iso.org/iso/catalogue_detail?csnumber=44169>.

[ISO_8601] International Organization for Standardization, "ISO/IEC International Standard 8601:2004; Data elements and interchange formats -- Information interchange -- Representation of dates and times", December 2004, <http://www.iso.org/iso/catalogue_detail?csnumber=40874>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4281] Gellens, R., Singer, D., and P. Frojdh, "The Codecs

Parameter for "Bucket" Media Types", RFC 4281,
November 2005.

11.2. Informative References

[M3U] Nullsoft, Inc., "The M3U Playlist format, originally invented for the Winamp media player", <http://wikipedia.org/wiki/M3U>.

Author's Address

Roger Pantos (editor)
Apple Inc.
Cupertino, California
United States

Email: http-live-streaming-review@group.apple.com