

Informational
Internet-Draft
Intended status: Informational
Expires: October 4, 2010

R. Pantos, Ed.
Apple Inc.
April 2, 2010

HTTP Live Streaming
draft-pantos-http-live-streaming-03

Abstract

This document describes a protocol for transferring unbounded streams of multimedia data. 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 2 of this protocol.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#). This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at

<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on October 4, 2010.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Pursuant to [Section 8.f.](#) of the Legal Provisions (see the IETF Trust's Legal Provisions Relating to IETF Documents effective December 28, 2009), [Section 4](#) of the Legal Provisions does not apply to this document. Thus, to the extent that this Informational Internet Draft contains one or more Code Components, no license to such Code Components is granted. Furthermore, this document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

Furthermore, this Informational Internet Draft is submitted as an RFC Editor Contribution and/or non-IETF Document (not as a Contribution, IETF Contribution, nor IETF Document) in accordance with [BCP 78](#) and [BCP 79](#).

Table of Contents

1.	Introduction	4
2.	Summary	4
3.	The Playlist file	4
3.1.	Introduction	4
3.2.	New Tags	5
3.2.1.	EXT-X-TARGETDURATION	6
3.2.2.	EXT-X-MEDIA-SEQUENCE	6
3.2.3.	EXT-X-KEY	6
3.2.4.	EXT-X-PROGRAM-DATE-TIME	7
3.2.5.	EXT-X-ALLOW-CACHE	7
3.2.6.	EXT-X-ENDLIST	7
3.2.7.	EXT-X-STREAM-INF	8
3.2.8.	EXT-X-DISCONTINUITY	8
3.2.9.	EXT-X-VERSION	9
4.	Media files	9
5.	Key files	10
5.1.	Introduction	10
5.2.	IV for AES-128	10
6.	Client/Server Actions	11
6.1.	Introduction	11
6.2.	Server Process	11
6.2.1.	Introduction	11
6.2.2.	Sliding Window Playlists	12
6.2.3.	Encrypting media files	13
6.2.4.	Providing variant streams	13
6.3.	Client Process	14
6.3.1.	Introduction	14
6.3.2.	Loading the Playlist file	14
6.3.3.	Playing the Playlist file	15
6.3.4.	Reloading the Playlist file	16
6.3.5.	Determining the next file to load	16
6.3.6.	Decrypting encrypted media files	17
7.	Protocol version compatibility	17
8.	Examples	17
8.1.	Introduction	17
8.2.	Simple Playlist file	17
8.3.	Sliding Window Playlist, using HTTPS	18
8.4.	Playlist file with encrypted media files	18
8.5.	Variant Playlist file	18
9.	Contributors	19
10.	IANA Considerations	19
11.	Security Considerations	20
12.	References	20
12.1.	Normative References	20
12.2.	Informative References	22
	Author's Address	22

Pantos

Expires October 4, 2010

[Page 3]

1. Introduction

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

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 media URIs and informational tags. Each media URI 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

3.1. Introduction

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 '#'. Blank lines are ignored. White space MUST NOT be present, except for elements in which it is explicitly specified.

A URI line identifies a media file or a variant Playlist file (see [Section 3.2.7](#)).

URIs MAY be relative. A relative URI MUST be resolved against the

URI of the Playlist file that contains it.

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.

The duration of a Playlist file is the sum of the durations of the media files within it.

M3U Playlist files whose names end in .m3u8 and/or have the HTTP Content-Type "application/vnd.apple.mpegurl" are encoded in UTF-8 [[RFC3629](#)]. Files whose names end with .m3u and/or have the HTTP Content-Type [[RFC2616](#)] "audio/mpegurl" are encoded in US-ASCII [[US_ASCII](#)].

Playlist files MUST have names that end in .m3u8 and/or have the Content-Type "application/vnd.apple.mpegurl" (if transferred over HTTP), or have names that end in .m3u and/or have the HTTP Content-Type type "audio/mpegurl" (for compatibility).

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, which is an optional human-readable informative title of the media segment.

This document defines the following 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, EXT-X-ENDLIST, EXT-X-DISCONTINUITY, and EXT-X-VERSION.

[3.2.](#) New Tags

3.2.1. EXT-X-TARGETDURATION

The EXT-X-TARGETDURATION tag specifies the maximum media file duration. The EXTINF duration of each media file in the Playlist file MUST be less than or equal to the target duration. This tag MUST appear once in the Playlist file. Its format is:

```
#EXT-X-TARGETDURATION:<s>
```

where s is an integer indicating the target duration in seconds.

3.2.2. 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>
```

A Playlist file MUST NOT contain more than one EXT-X-MEDIA-SEQUENCE tag. 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 0.

A media file's sequence number is not required to appear in its URI.

See [Section 6.3.2](#) and [Section 6.3.5](#) for information on handling the EXT-X-MEDIA-SEQUENCE tag.

3.2.3. 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>"][,IV=<IV>]
```

The METHOD attribute specifies the encryption method. Two encryption methods are defined: NONE and AES-128.

An encryption method of NONE means that media files are not encrypted. If the encryption method is NONE, the URI and the IV attributes MUST NOT be present.

An encryption method of AES-128 means that media files are encrypted using the Advanced Encryption Standard [[AES 128](#)] with a 128-bit key and PKCS7 padding [[RFC5652](#)]. If the encryption method is AES-128,

the URI attribute MUST be present. The IV attribute MAY be present; see [Section 5.2](#).

The URI attribute, if present, specifies how to obtain the key. The IV attribute, if present, specifies the Initialization Vector to be used with the key. The IV attribute appeared in protocol version 2.

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

If the Playlist file does not contain an EXT-X-KEY tag then media files are not encrypted.

See [Section 5](#) for the format of the key file, and [Section 5.2](#), [Section 6.2.3](#) and [Section 6.3.6](#) for additional information on media file encryption.

[3.2.4](#). 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>
```

See [Section 6.2.1](#) and [Section 6.3.3](#) for more information on the EXT-X-PROGRAM-DATE-TIME tag.

[3.2.5](#). EXT-X-ALLOW-CACHE

The EXT-X-ALLOW-CACHE tag indicates whether the client MAY or MUST NOT cache downloaded media files for later replay. It MAY occur anywhere in the Playlist file; it MUST NOT occur more than once. The EXT-X-ALLOW-CACHE tag applies to all segments in the playlist. Its format is:

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

See [Section 6.3.3](#) for more information on the EXT-X-ALLOW-CACHE tag.

[3.2.6](#). EXT-X-ENDLIST

The EXT-X-ENDLIST tag indicates that no more media files will be added to the Playlist file. It MAY occur anywhere in the Playlist file; it MUST NOT occur more than once. Its format is:

```
#EXT-X-ENDLIST
```


3.2.7. 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>
```

An attribute MUST NOT occur more than once in the same EXT-X-STREAM-INF tag. The following attributes are defined:

BANDWIDTH=<n>

where n is a number of bits per second. It MUST be an upper bound of the overall bitrate of each media file, calculated to include container overhead, that appears or will appear in the Playlist.

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 tags with the same PROGRAM-ID to identify different encodings of the same presentation. These variant playlists MAY contain additional EXT-X-STREAM-INF tags.

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](#)].

RESOLUTION=<N>x<M>

where N is the approximate encoded horizontal resolution of video within the stream, expressed as a number of pixels, and M is the approximate encoded vertical resolution.

3.2.8. EXT-X-DISCONTINUITY

The EXT-X-DISCONTINUITY tag indicates an encoding discontinuity between the media file that follows it and the one that preceded it. The set of characteristics that MAY change is:

- o file format
- o number and type of tracks
- o encoding parameters
- o encoding sequence
- o timestamp sequence

Its format is:

#EXT-X-DISCONTINUITY

See [Section 4](#), [Section 6.2.1](#), and [Section 6.3.3](#) for more information about the EXT-X-DISCONTINUITY tag.

3.2.9. EXT-X-VERSION

The EXT-X-VERSION tag indicates the compatibility version of the Playlist file. The Playlist file, its associated media, and its server MUST comply with all provisions of the most-recent version of this document describing the protocol version indicated by the tag value.

Its format is:

#EXT-X-VERSION:<n>

where n is an integer indicating the protocol version.

A Playlist file MUST NOT contain more than one EXT-X-VERSION tag. A Playlist file that does not contain an EXT-X-VERSION tag MUST comply with version 1 of this protocol.

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 or an MPEG-2 audio elementary stream [[ISO 13818](#)].

Transport Stream files MUST contain a single MPEG-2 Program. There SHOULD be a Program Association Table and a Program Map Table at the start of each file. A file that contains video SHOULD have at least one key frame and enough information to completely initialize a video decoder.

A media file in a Playlist MUST be the continuation of the encoded stream at the end of the media file with the previous sequence number unless it was the first media file to appear in the Playlist file or if it is preceded by an EXT-X-DISCONTINUITY tag.

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

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 accommodate a resolution change.

5. Key files

5.1. Introduction

An EXT-X-KEY tag with the URI attribute 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.

5.2. 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.

If the EXT-X-KEY tag has the IV attribute, implementations MUST use the attribute value as the IV when encrypting or decrypting with that key. The value MUST be interpreted as a 128-bit hexadecimal number and MUST be prefixed with 0x or 0X.

If the EXT-X-KEY tag does not have the IV attribute, implementations MUST use the sequence number of the media file as the IV when encrypting or decrypting that media file. 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

6.1. Introduction

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

6.2. Server Process

6.2.1. Introduction

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 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 maximum EXTINF value of any media file added to the Playlist file. Its value **MUST** remain constant for the entire presentation. A typical target duration is 10 seconds.

The Playlist file **SHOULD** contain one EXT-X-VERSION tag which indicates the compatibility version of the stream. Its value **SHOULD** be the lowest protocol version with which the server, Playlist file, and associated media files all comply.

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.

The server **MUST NOT** change the value of the EXT-X-ALLOW-CACHE tag.

Every media file URI in a Playlist **MUST** be prefixed with an EXTINF

tag indicating the rounded 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 provides an informative mapping of the timeline of the media to an appropriate wall-clock time, which may be used as a basis for seeking, for display, or for other purposes. If a server provides this mapping, it SHOULD place an EXT-X-PROGRAM-DATE-TIME tag after every EXT-X-DISCONTINUITY tag in the Playlist file.

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

If the Playlist does not contain the EXT-X-ENDLIST tag, the server MUST make a new version of the Playlist file available that contains at least one new media file URI. It MUST be made available relative to the time that the previous version of the Playlist file was made available: no earlier than one-half the target duration after that time, and no later than 1.5 times the target duration after that time.

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.

6.2.2. 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 SHOULD 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.

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

The duration of a Playlist file that does not contain the EXT-X-ENDLIST tag MUST be at least three times the target duration.

6.2.3. 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 HTTP Expires header in the key response to indicate that the key may be cached.

If the encryption METHOD is AES-128, AES-128 CBC encryption SHALL be applied to individual media files. The entire file MUST be encrypted. Cipher Block Chaining MUST NOT be applied across media files. The IV used for encryption MUST be either the sequence number of the media file or the value of the IV attribute of the EXT-X-KEY tag, as described in [Section 5.2](#).

The server MUST encrypt every media file in a Playlist using the method and other attributes specified by the EXT-X-KEY tag that most immediately precedes its URI in the Playlist file. Media files preceded by an EXT-X-KEY tag whose METHOD is NONE, or not preceded by any EXT-X-KEY tag, MUST NOT be encrypted.

The URI of every EXT-X-KEY tag must be distinct from the URI of every other EXT-X-KEY tag that appears or has appeared in the Playlist file, unless its METHOD is NONE. An EXT-X-KEY tag with a METHOD of NONE MUST NOT contain a URI attribute.

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.

6.2.4. 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** present the same content, including stream discontinuities.

Each variant Playlist file **MUST** have the same target duration.

Content that appears in one variant Playlist file but not in another **MUST** appear either at the beginning or at the end of the Playlist file and **MUST NOT** be longer than the target duration.

Matching content in variant streams **MUST** have matching timestamps. This allows clients to synchronize the streams.

Elementary Audio Stream files **MUST** signal the timestamp of the first sample in the file by prepending an ID3 PRIV tag [[ID3](#)] with an owner identifier of "com.apple.streaming.transportStreamTimestamp". The binary data **MUST** be a 33-bit MPEG-2 Program Elementary Stream timestamp expressed as a big-endian eight-octet number.

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

[6.3.](#) Client Process

[6.3.1.](#) Introduction

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.

[6.3.2.](#) Loading the Playlist file

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

The client **MUST** ensure that the Playlist file begins with the EXTM3U tag and that the EXT-X-VERSION tag, if present, specifies a protocol version supported by the client; if not, the client **MUST**

NOT attempt to use the Playlist.

The client SHOULD ignore any tags and attributes it does not recognize.

The client MUST determine the next media file to load as described in [Section 6.3.5](#).

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.

6.3.3. Playing the Playlist file

The client SHALL choose which media file to play first from the Playlist when playback starts. If the EXT-X-ENDLIST tag is not present and the client intends to play the media regularly (i.e. in playlist order at the nominal playback rate), the client SHOULD NOT choose a file which starts less than three target durations from the end of the Playlist file. Doing so can trigger playback stalls.

To achieve regular playback, media files MUST be played in the order that they appear in the Playlist file. The client MAY present the available media in any way it wishes, including regular playback, random access, and trick modes.

The client MUST be prepared to reset its parser(s) and decoder(s) before playing a media file that is preceded by an EXT-X-DISCONTINUITY tag.

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.

6.3.4. 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. 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 a multiple of the target duration. This multiple is 0.5 for the first attempt, 1.5 for the second, and 3.0 thereafter.

6.3.5. 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 6.3.3](#).

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.

6.3.6. Decrypting 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 IV used for decryption MUST be either the sequence number of the media file or the value of the IV attribute of the EXT-X-KEY tag, as described in [Section 5.2](#).

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. Protocol version compatibility

Clients and servers MUST implement protocol version 2 or higher to use:

- o The IV attribute of the EXT-X-KEY tag.

8. Examples

8.1. Introduction

This section contains several example Playlist files.

8.2. Simple Playlist file

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


8.3. 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
```

8.4. 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/fileSequence52-1.ts
#EXTINF:15,
http://media.example.com/fileSequence52-2.ts
#EXTINF:15,
http://media.example.com/fileSequence52-3.ts

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

#EXTINF:15,
http://media.example.com/fileSequence53-1.ts
```

8.5. 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
```


9. Contributors

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

10. IANA Considerations

This memo requests that the following MIME type [[RFC2046](#)] be registered with the IANA:

Type name: "application"

Subtype name: "vnd.apple.mpegurl"

Required parameters: (none)

Optional parameters: (none)

Encoding considerations: encoded as text. See [Section 3](#) for more information.

Security considerations: See [Section 11](#).

Compression: this media type does not employ compression.

Interoperability considerations: There are no byte-ordering issues, since files are 7- or 8-bit text. Applications could encounter unrecognized tags, which SHOULD be ignored.

Published specification: see [Section 3](#).

Applications that use this media type: Multimedia applications such as the iPhone media player (OS 3.0) and QuickTime Player in Mac OS X Snow Leopard.

Additional information: files begin with the magic number #EXTM3U. Filenames normally end with .m3u8 or .m3u (see [Section 3](#)). No Macintosh file type codes have been registered.

Person & email address to contact for further information: David Singer, [singer AT apple.com](#).

Intended usage: LIMITED USE

Restrictions on usage: (none)

Author: Roger Pantos

Change Controller: David Singer

11. Security Considerations

Since the protocol generally uses HTTP to transfer data, 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.

Playlist files contain URIs, which clients will use to make network requests of arbitrary entities. Clients SHOULD range-check responses to prevent buffer overflows. See also the Security Considerations section of [RFC 3986](#) [[RFC3986](#)].

Clients SHOULD load resources identified by URI lazily to avoid contributing to denial-of-service attacks.

HTTP requests often include session state ("cookies"), which may contain private user data. Implementations MUST follow cookie restriction and expiry rules specified by [RFC 2965](#) [[RFC2965](#)]. See also the Security Considerations section of [RFC 2965](#), and [RFC 2964](#) [[RFC2964](#)].

Encryption keys are specified by URI. The delivery of these keys SHOULD be secured by a mechanism such as HTTP over TLS [[RFC5246](#)] (formerly SSL) in conjunction with a secure realm or a session cookie.

12. References

12.1. 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", October 2007, <http://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.
- [RFC2964] Moore, K. and N. Freed, "Use of HTTP State Management", [BCP 44](#), [RFC 2964](#), October 2000.
- [RFC2965] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", [RFC 2965](#), October 2000.
- [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.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", [RFC 5652](#), September 2009.
- [US_ASCII] American National Standards Institute, "ANSI X3.4-1986, Information Systems -- Coded Character Sets 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)", December 1986.

12.2. Informative References

- [ID3] ID3.org, "The ID3 audio file data tagging format",
<http://www.id3.org/Developer_Information>.

- [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

