HTTP Live Streaming
draft-pantos-http-live-streaming-08

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 4 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   and derivative works of it may not be created, and it may not be
   published except as an Internet-Draft.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). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   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."

   This Internet-Draft will expire on September 24, 2012.

Copyright Notice

   This Informational Internet Draft is submitted as an RFC Editor

Table of Contents

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. The URIs and their associated tags specify a
series of media segments.

To play the stream, the client first obtains the Playlist file and
then obtains and plays each media segment 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 further 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, blank, or
starts with the 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 segment or a variant Playlist file (see
Section 3.4.10). Each media segment is specified by a media URI and
the tags that apply to it.

   Lines that start with the character '#' are either comments or tags.

   Tags begin with #EXT. All other lines that begin with '#' are
   comments and SHOULD be ignored.

   A URI in a Playlist, whether it is a URI line or part of a tag, MAY
   be relative. Relative URIs MUST be resolved against the URI of the
   Playlist file that contains it.

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

   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).

3.2. Attribute Lists

   Certain extended M3U tags have values which are Attribute Lists. An
   Attribute List is a comma-separated list of attribute/value pairs
   with no whitespace.

   An attribute/value pair has the following syntax:

   AttributeName=AttributeValue

   An AttributeName is an unquoted string containing characters from the
   set [A..Z] and '-'.

   An AttributeValue is one of the following:

   o decimal-integer: an unquoted string of characters from the set
      [0..9] expressing an integer in base-10 arithmetic.

   o hexadecimal-integer: an unquoted string of characters from the set
      [0..9] and [A..F] that is prefixed with 0x or 0X and which
      expresses an integer in base-16 arithmetic.

   o decimal-floating-point: an unquoted string of characters from the
      set [0..9] and '.' which expresses a floating-point number in
      decimal positional notation.

o quoted-string: a string of characters within a pair of double-
   quotes ("). The set of characters allowed in the string and any
   rules for escaping special characters are specified by the
   Attribute definition, but any double-quote (") character and any
   carriage-return or linefeed will always be replaced by an escape
   sequence.

o enumerated-string: an unquoted character string from a set which
   is explicitly defined by the Attribute. An enumerated-string will
   never contain double-quotes ("), commas (,), or whitespace.

o decimal-resolution: two decimal-integers separated by the "x"
   character. The first integer is a horizontal pixel dimension
   (width); the second is a vertical pixel dimension (height).

The type of the AttributeValue for a given AttributeName is specified
by the Attribute definition.

A given AttributeName MUST NOT appear more than once in a given
Attribute List.

An Attribute/value pair with an unrecognized AttributeName MUST be
ignored by the client.

Attribute/value pairs of type enumerated-string that contain
unrecognized values SHOULD be ignored by the client.

3.3. Standard Tags

3.3.1. EXTM3U

An Extended M3U file is distinguished from a basic M3U file by its
first line, which MUST be the tag #EXTM3U.

3.3.2. EXTINF

The EXTINF tag specifies the duration of a media segment. It applies
only to the media URI that follows it. Each media segment URI MUST
be preceded by an EXTINF tag. Its format is:

#EXTINF:<duration>,<title>

"duration" is an integer or floating-point number in decimal
positional notation that specifies the duration of the media segment
in seconds. Durations that are reported as integers SHOULD be
rounded to the nearest integer. Durations MUST be integers if the
protocol version of the Playlist file is less than 3. The remainder
of the line following the comma is an optional human-readable

informative title of the media segment.

3.4. New Tags

This document defines the following new tags: EXT-X-BYTERANGE, EXT-X-
TARGETDURATION, EXT-X-MEDIA-SEQUENCE, EXT-X-KEY, EXT-X-PROGRAM-DATE-
TIME, EXT-X-ALLOW-CACHE, EXT-X-PLAYLIST-TYPE, EXT-X-STREAM-INF, EXT-
X-I-FRAME-STREAM-INF, EXT-X-I-FRAMES-ONLY, EXT-X-MEDIA, EXT-X-
ENDLIST, EXT-X-DISCONTINUITY, and EXT-X-VERSION.

3.4.1. EXT-X-BYTERANGE

The EXT-X-BYTERANGE tag indicates that a media segment is a sub-range
of the resource identified by its media URI. It applies only to the
next media URI that follows it in the Playlist. Its format is:

#EXT-X-BYTERANGE:<n>[@o]

where n is a decimal-integer indicating the length of the sub-range
in bytes. If present, o is a decimal-integer indicating the start of
the sub-range, as a byte offset from the beginning of the resource.
If o is not present, the sub-range begins at the next byte following
the sub-range of the previous media segment.

If o is not present, a previous media segment MUST appear in the
Playlist file and MUST be a sub-range of the same media resource.

A media URI with no EXT-X-BYTERANGE tag applied to it specifies a
media segment that consists of the entire resource.

The EXT-X-BYTERANGE tag appeared in version 4 of the protocol.

3.4.2. EXT-X-TARGETDURATION

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

#EXT-X-TARGETDURATION:<s>

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

3.4.3. EXT-X-MEDIA-SEQUENCE

Each media URI in a Playlist has a unique integer 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 URI is not required to contain its sequence number.

See Section 6.3.2 and Section 6.3.5 for information on handling the
EXT-X-MEDIA-SEQUENCE tag.

3.4.4. EXT-X-KEY

Media segments MAY be encrypted. The EXT-X-KEY tag specifies how to
decrypt them. It applies to every media URI that appears between it
and the next EXT-X-KEY tag in the Playlist file (if any). Its format
is:

    #EXT-X-KEY:<attribute-list>

The following attributes are defined:

The METHOD attribute specifies the encryption method. It is of type
enumerated-string. Each EXT-X-KEY tag MUST contain a METHOD
attribute.

Two methods are defined: NONE and AES-128.

An encryption method of NONE means that media segments 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 segments 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 specifies how to obtain the key. Its value is a

quoted-string that contains a URI [RFC3986] for the key.

The IV attribute, if present, specifies the Initialization Vector to
be used with the key. Its value is a hexadecimal-integer. The IV
attribute appeared in protocol version 2.

If the Playlist file does not contain an EXT-X-KEY tag then media
segments 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
segment encryption.

3.4.5. EXT-X-PROGRAM-DATE-TIME

The EXT-X-PROGRAM-DATE-TIME tag associates the first sample of a
media segment with an absolute date and/or time. It applies only to
the next media URI.

The date/time representation is ISO/IEC 8601:2004 [ISO_8601] and
SHOULD indicate a time zone:

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

For example:

#EXT-X-PROGRAM-DATE-TIME:2010-02-19T14:54:23.031+08:00

See Section 6.2.1 and Section 6.3.3 for more information on the EXT-
X-PROGRAM-DATE-TIME tag.

3.4.6. EXT-X-ALLOW-CACHE

The EXT-X-ALLOW-CACHE tag indicates whether the client MAY or MUST
NOT cache downloaded media segments 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.

3.4.7. EXT-X-PLAYLIST-TYPE

The EXT-X-PLAYLIST-TYPE tag provides mutability information about the
Playlist file. It applies to the entire Playlist file. It is
optional. Its format is:

   #EXT-X-PLAYLIST-TYPE:<EVENT|VOD>

Section 6.2.1 defines the implications of the EXT-X-PLAYLIST-TYPE
tag.

3.4.8. EXT-X-ENDLIST

The EXT-X-ENDLIST tag indicates that no more media segments 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

3.4.9. EXT-X-MEDIA

The EXT-X-MEDIA tag is used to relate Playlists that contain
alternative renditions of the same content. For example, three EXT-
X-MEDIA tags can be used to identify audio-only Playlists that
contain English, French and Spanish renditions of the same
presentation. Or two EXT-X-MEDIA tags can be used to identify video-
only Playlists that show two different camera angles.

The EXT-X-MEDIA tag stands alone, in that it does not apply to a
particular URI in the Playlist. Its format is:

   #EXT-X-MEDIA:<attribute-list>

The following attributes are defined:

   URI

The value is a quoted-string containing a URI that identifies the
Playlist file. This attribute is optional; see Section 3.4.10.1.

   TYPE

The value is enumerated-string; valid strings are AUDIO and VIDEO.
If the value is AUDIO, the Playlist described by the tag MUST contain
audio media. If the value is VIDEO, the Playlist MUST contain video
media.

   GROUP-ID

   The value is a quoted-string identifying a mutually-exclusive group
   of renditions. The presence of this attribute signals membership in
   the group. See Section 3.4.9.1.

   LANGUAGE

   The value is a quoted-string containing an RFC 5646 [RFC5646]
   language tag that identifies the primary language used in the
   rendition. This attribute is optional.

   NAME

   The value is a quoted-string containing a human-readable description
   of the rendition. If the LANGUAGE attribute is present then this
   description SHOULD be in that language.

   DEFAULT

   The value is enumerated-string; valid strings are YES and NO. If the
   value is YES, then the client SHOULD play this rendition of the
   content in the absence of information from the user indicating a
   different choice. This attribute is optional. Its absence indicates
   an implicit value of NO.

   AUTOSELECT

   The value is enumerated-string; valid strings are YES and NO. This
   attribute is optional. Its absence indicates an implicit value of
   NO. If the value is YES, then the client MAY choose to play this
   rendition in the absence of explicit user preference because it
   matches the current playback environment, such as chosen system
   language.

   The EXT-X-MEDIA tag appeared in version 4 of the protocol.

3.4.9.1. Rendition Groups

   A set of EXT-X-MEDIA tags with the same GROUP-ID value forms a group
   of renditions. Each member of the group MUST represent an
   alternative rendition of the same content.

   All EXT-X-MEDIA tags in a Playlist MUST meet the following
   constraints:

o All EXT-X-MEDIA tags in the same group MUST have the same TYPE
   attribute.

o All EXT-X-MEDIA tags in the same group MUST have different NAME
   attributes.

o A group MUST NOT have more than one member with a DEFAULT
   attribute of YES.

o All members of a group whose AUTOSELECT attribute has a value of
   YES MUST have LANGUAGE [RFC5646] attributes with unique values.

o All members of a group with TYPE=AUDIO MUST use the same audio
   sample format.

o All members of a group with TYPE=VIDEO MUST use the same video
   sample format.

A Playlist MAY contain multiple groups of the same TYPE in order to
provide multiple encodings of each rendition. If it does so, each
group of the same TYPE SHOULD contain corresponding members with the
same NAME attribute, LANGUAGE attribute, and rendition.

3.4.10. EXT-X-STREAM-INF

The EXT-X-STREAM-INF tag identifies a media URI as a Playlist file
containing a multimedia presentation and provides information about
that presentation. It applies only to the URI that follows it. Its
format is:

#EXT-X-STREAM-INF:<attribute-list>
<URI>

The following attributes are defined:

BANDWIDTH

The value is a decimal-integer of bits per second. It MUST be an
upper bound of the overall bitrate of each media segment (calculated
to include container overhead) that appears or will appear in the
Playlist.

Every EXT-X-STREAM-INF tag MUST include the BANDWIDTH attribute.

PROGRAM-ID

The value is a decimal-integer 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

The value is a quoted-string containing a comma-separated list of
formats, where each format specifies a media sample type that is
present in a media segment in the Playlist file. Valid format
identifiers are those in the ISO File Format Name Space defined by
RFC 6381 [RFC6381].

Every EXT-X-STREAM-INF tag SHOULD include a CODECS attribute.

RESOLUTION

The value is a decimal-resolution describing the approximate encoded
horizontal and vertical resolution of video within the presentation.

AUDIO

The value is a quoted-string. It MUST match the value of the
GROUP-ID attribute of an EXT-X-MEDIA tag elsewhere in the Playlist
whose TYPE attribute is AUDIO. It indicates the set of audio
renditions that MAY be used when playing the presentation. See
Section 3.4.10.1.

VIDEO

The value is a quoted-string. It MUST match the value of the
GROUP-ID attribute of an EXT-X-MEDIA tag elsewhere in the Playlist
whose TYPE attribute is VIDEO. It indicates the set of video
renditions that MAY be used when playing the presentation. See
Section 3.4.10.1.

3.4.10.1. Alternative Renditions

When an EXT-X-STREAM-INF tag contains an AUDIO or a VIDEO attribute,
it indicates that alternative renditions of the content are available
for playback of that variant.

When defining alternative renditions, the following constraints MUST
be met:

o All playable combinations of renditions associated with an EXT-X-
   STREAM-INF tag MUST have an aggregate bandwidth less than or equal
   to the BANDWIDTH attribute of the EXT-X-STREAM-INF tag.

o If an EXT-X-STREAM-INF tag contains a RESOLUTION attribute and a
   VIDEO attribute, then every alternative video rendition MUST match
   the value of the RESOLUTION attribute.

o Every alternative rendition associated with an EXT-X-STREAM-INF
   tag MUST meet the constraints for a variant stream described in
   Section 6.2.4.

The URI attribute of an EXT-X-MEDIA tag is optional. If it is
missing, it indicates that the rendition described by the EXT-X-MEDIA
tag is present in the main Playlist described by the EXT-X-STREAM-INF
tag.

Note that if a client chooses to play renditions of audio and video
that are not present in the main Playlist described by the EXT-X-
STREAM-INF tag, or if the client chooses to play an audio rendition
and the main Playlist is audio-only, then the client MAY ignore the
main Playlist and its media.

3.4.11. EXT-X-DISCONTINUITY

The EXT-X-DISCONTINUITY tag indicates an encoding discontinuity
between the media segment 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.

3.4.12. EXT-X-I-FRAMES-ONLY

The EXT-X-I-FRAMES-ONLY tag indicates that each media segment in the
Playlist describes a single I-frame. I-frames (or Intra frames) are
encoded video frames whose encoding does not depend on any other
frame.

The EXT-X-I-FRAMES-ONLY tag applies to the entire Playlist. Its
format is:

#EXT-X-I-FRAMES-ONLY

In a Playlist with the EXT-X-I-FRAMES-ONLY tag, the media segment
duration (EXTINF tag value) is the time between the presentation time
of the I-frame in the media segment and the presentation time of the
next I-frame in the Playlist, or the end of the presentation if it is
the last I-frame in the Playlist.

Media resources containing I-frame segments MUST begin with a
Transport Stream PAT/PMT. The byte range of an I-frame segment with
an EXT-X-BYTERANGE tag applied to it (Section 3.4.1) MUST NOT include
a PAT/PMT.

The EXT-X-I-FRAMES-ONLY tag appeared in version 4 of the protocol.

3.4.13. EXT-X-I-FRAME-STREAM-INF

The EXT-X-I-FRAME-STREAM-INF tag identifies a Playlist file
containing the I-frames of a multimedia presentation. It stands
alone, in that it does not apply to a particular URI in the Playlist.
Its format is:

#EXT-X-I-FRAME-STREAM-INF:<attribute-list>

All attributes defined for the EXT-X-STREAM-INF tag (Section 3.4.10)
are also defined for the EXT-X-I-FRAME-STREAM-INF tag, except for the
AUDIO attribute. In addition, the following attribute is defined:

URI

The value is a quoted-string containing a URI that identifies the
I-frame Playlist file.

Every EXT-X-I-FRAME-STREAM-INF tag MUST include a BANDWIDTH attribute
and a URI attribute.

The provisions in Section 3.4.10.1 also apply to EXT-X-I-FRAME-
STREAM-INF tags with a VIDEO attribute.

A Playlist that specifies alternative VIDEO renditions and I-frame
Playlists SHOULD include an alternative I-frame VIDEO rendition for
each regular VIDEO rendition, with the same NAME and LANGUAGE
attributes.

The EXT-X-I-FRAME-STREAM-INF tag appeared in version 4 of the

protocol. Clients that do not implement protocol version 4 or higher
MUST ignore it.

3.4.14. 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.

The EXT-X-VERSION tag applies to the entire Playlist file. 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 segments

Each media URI in a Playlist file specifies a media segment which is
part of the overall presentation. If a media URI has an EXT-X-
BYTERANGE tag applied to it, the segment is a sub-range of the media
file identified by the URI. Otherwise, the segment is the entire
media file.

Each media segment MUST be formatted as an MPEG-2 Transport Stream or
an MPEG-2 audio elementary stream [ISO_13818].

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

A media segment MUST be the continuation of the encoded media at the
end of the segment with the previous sequence number, where values in
a continuous series, such as timestamps and Continuity Counters,
continue uninterrupted - unless the media segment was the first ever
to appear in the Playlist file or has an EXT-X-DISCONTINUITY tag
applied to it.

Clients SHOULD be prepared to handle multiple tracks of a particular

   type (e.g. audio or video). A client with no other preference SHOULD
   choose the track with the lowest numerical PID that it can play.

   Clients MUST ignore private streams inside Transport Streams that
   they do not recognize.

   The encoding parameters for samples in a stream inside a media
   segment and between corresponding streams across multiple media
   segments 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 a URI attribute identifies a Key file. A Key
   file contains the cipher key that MUST be used to decrypt subsequent
   media segments in the Playlist.

   The AES-128 encryption method uses 16-octet keys. The format of the
   Key file is a single packed array of 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 segment as the IV when
   encrypting or decrypting that media segment. 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 segments 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 segments
   whose duration is less than or equal to a constant target duration.
   The server SHOULD attempt to divide the stream at points that support
   effective decode of individual media segments, e.g. on packet and key
   frame boundaries.

   The server MUST create a URI for every media segment that enables its
   clients to obtain the segment data. If a server supports partial
   loading of resources (e.g. via HTTP Range requests), it MAY specify
   segments as sub-ranges of larger resources using the EXT-X-BYTERANGE
   tag.

   The server MUST create a Playlist file. The Playlist file MUST
   conform to the format described in Section 3. A URI for each media
   segment 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
   segment MUST be available to clients if its URI is in the Playlist
   file.

   The Playlist file MUST contain an EXT-X-TARGETDURATION tag. Its
   value MUST be equal to or greater than the EXTINF value of any media
   segment that appears or will appear in the Playlist file. Its value
   MUST NOT change. 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 MUST be
   the lowest protocol version with which the server, Playlist file, and
   associated media segments all comply. Its value MUST NOT change.

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

   If the Playlist file is distributed by HTTP, the server SHOULD
   support client requests to use "gzip" Content-Encoding.

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

   The server MUST NOT change the Playlist file, except to:

Append lines to it (Section 6.2.1).

Remove media URIs from the Playlist in the order that they appear, along with any tags that apply only to those media URIs (Section 6.2.2).

Increment the value of the EXT-X-MEDIA-SEQUENCE tag (Section 6.2.2).

Add or remove EXT-X-STREAM-INF tags or EXT-X-I-FRAME-STREAM-INF tags (Section 6.2.4). Note that clients are not required to reload variant Playlist files, so changing them may not have immediate effect.

Add an EXT-X-ENDLIST tag to the Playlist (Section 6.2.1).

Furthermore, the Playlist file MAY contain an EXT-X-PLAYLIST-TYPE tag with a value of either EVENT or VOD. If the tag is present and has a value of EVENT, the server MUST NOT change or delete any part of the Playlist file (although it MAY append lines to it). If the tag is present and has a value of VOD, the Playlist file MUST NOT change.

Every media URI in a Playlist MUST have an EXTINF tag applied to it indicating the duration of the media segment.

The server MAY associate an absolute date and time with a media segment by applying an EXT-X-PROGRAM-DATE-TIME tag to its URI. The date and time value 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 apply an EXT-X-PROGRAM-DATE-TIME tag to every segment that has an EXT-X-DISCONTINUITY tag applied to it.

If the Playlist contains the final media segment 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 segment. 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 segments 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 segments by removing
   media URIs from the Playlist file. If media URIs are to be removed,
   the Playlist file MUST contain exactly one EXT-X-MEDIA-SEQUENCE tag.
   Its value MUST be incremented by 1 for every media URI that is
   removed from the Playlist file.

   Media URIs MUST be removed from the Playlist file in the order that
   they appear in the Playlist.

   The server MUST NOT remove a media URI specifying a segment from the
   Playlist file if the duration of the Playlist file minus the duration
   of the segment is less than three times the target duration.

   When the server removes a media URI from the Playlist, the
   corresponding media segment SHOULD remain available to clients for a
   period of time equal to the duration of the segment plus the duration
   of the longest Playlist file distributed by the server containing
   that segment.

   If a server plans to remove a media segment 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.

6.2.3. Encrypting media segments

   If media segments 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.

   The server MUST encrypt every media segment in a Playlist according
   to the EXT-X-KEY tag that applies to its URI in the Playlist file.
   Media segments with an EXT-X-KEY tag whose METHOD is NONE, or which
   do not have an EXT-X-KEY tag applied to them, MUST NOT be encrypted.

   If the encryption METHOD is AES-128 and the Playlist does not contain
   the EXT-X-I-FRAMES-ONLY tag, AES-128 CBC encryption with PKCS7
   padding [RFC5652] SHALL be applied to individual media segments. The
   entire segment MUST be encrypted. Cipher Block Chaining MUST NOT be
   applied across media segments. The IV used for encryption MUST be
   either the sequence number of the media segment or the value of the
   IV attribute of the EXT-X-KEY tag, as described in Section 5.2.

If the encryption METHOD is AES-128 and the Playlist contains an EXT-
X-I-FRAMES-ONLY tag, AES-128 CBC encryption with PKCS7 padding
[RFC5652] MUST be applied to the entire media resource. The entire
resource MUST be encrypted. Encryption MAY be restarted on 16-byte
block boundaries, unless the first block contains an I-frame. The IV
used for encryption MUST be either the sequence number of the media
segment or the value of the IV attribute of the EXT-X-KEY tag, as
described in Section 5.2.

The server MUST NOT remove an EXT-X-KEY tag from the Playlist file if
it applies to any media URI in the Playlist file.

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 or EXT-X-I-
FRAME-STREAM-INF tag for each variant stream. Each tag identifying
an encoding of 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 or EXT-X-I-FRAME-STREAM-INF tag contains
the CODECS attribute, the attribute value MUST include every format
defined by [RFC6381] that is present in any media segment 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.

   Each Elementary Audio Stream segment MUST signal the timestamp of
   its first sample with an ID3 PRIV tag [ID3] at the beginning of
   the segment. The ID3 PRIV owner identifier MUST be

"com.apple.streaming.transportStreamTimestamp". The ID3 payload
MUST be a 33-bit MPEG-2 Program Elementary Stream timestamp
expressed as a big-endian eight-octet number, with the upper 31
bits set to zero.

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 segment 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 segment in it will become unavailable
at the time that the Playlist file was loaded plus the duration of
the Playlist file.

6.3.3. Playing the Playlist file

The client SHALL choose which media segment 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 segment 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 segments 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 segment that has an EXT-X-DISCONTINUITY tag
applied to it.

The client SHOULD attempt to load media segments 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 segments after they
have been played. Otherwise the client MAY cache downloaded media
segments 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

segment in the Playlist. Media segment 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 one-half the target
duration before retrying.

In order to reduce server load, the client SHOULD NOT reload the
Playlist files of variant streams that are not currently being
played. If it decides to switch playback to a different variant, it
SHOULD stop reloading the Playlist of the old variant and begin
loading the Playlist of the new variant. It can use the EXTINF
durations and the constraints in Section 6.2.4 to determine the
approximate location of corresponding media. Once media from the new
variant has been loaded, the timestamps in the media segments can be
used to synchronize the old and new timelines precisely.

6.3.5. Determining the next segment to load

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

The first segment to load MUST be the segment that the client has
chosen to play first, as described in Section 6.3.3.

If the first segment 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 segment at the offset it was originally found at,
halting playback if it does not. The next media segment to load MUST
be the first media URI following the last-loaded URI in the Playlist.

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

6.3.6. Decrypting encrypted media segments

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 segments to which that EXT-X-KEY tag applies.

If the encryption METHOD is AES-128, AES-128 CBC decryption SHALL be
applied to individual media segments. The entire segment MUST be
decrypted. Cipher Block Chaining MUST NOT be applied across media
segments. The IV used for decryption MUST be either the sequence
number of the media segment or the value of the IV attribute of the

EXT-X-KEY tag, as described in Section 5.2.

If the media segment is part of an I-frame playlist (Section 3.4.12)
special care MUST be taken in loading and decrypting the segment,
because the resource identified by the URI is encrypted in 16-byte
blocks from offset 0. The sub-range specified by the EXT-X-BYTERANGE
tag MUST be widened to include the 16-byte blocks in which the
beginning and end of the sub-range fall. Next, it MUST be widened
further to include the previous 16-byte block. That range MUST be
loaded and decrypted with AES-128 CBC using an arbitrary IV. The
decrypted segment will then be the sub-range specified by the EXT-X-
BYTERANGE tag.

An EXT-X-KEY tag with a METHOD of NONE indicates that the media
segments it applies to are not encrypted.

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.

Clients and servers MUST implement protocol version 3 or higher to
use:

o Floating-point EXTINF duration values.

Clients and servers MUST implement protocol version 4 or higher to
use:

o The EXT-X-BYTERANGE tag.

o The EXT-X-I-FRAME-STREAM-INF tag.

o The EXT-X-I-FRAMES-ONLY tag.

o The EXT-X-MEDIA tag.

o The AUDIO and VIDEO attributes of the EXT-X-STREAM-INF 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 segments

```
#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
```

8.6. Variant Playlist with I-Frames

In this example, the PROGRAM-ID attributes have been left out:

```
#EXTM3U
#EXT-X-STREAM-INF:BANDWIDTH=1280000
low/audio-video.m3u8
#EXT-X-I-FRAME-STREAM-INF:BANDWIDTH=86000,URI="low/iframe.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=2560000
mid/audio-video.m3u8
#EXT-X-I-FRAME-STREAM-INF:BANDWIDTH=150000,URI="mid/iframe.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=7680000
hi/audio-video.m3u8
#EXT-X-I-FRAME-STREAM-INF:BANDWIDTH=550000,URI="hi/iframe.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=65000,CODECS="mp4a.40.5"
audio-only.m3u8
```

8.7. Variant Playlist with Alternative audio

In this example, the PROGRAM-ID attributes have been left out. The
CODECS attributes have been condensed for space. A '\' is used to
indicate that the tag continues on the following line with whitespace
removed:

```
#EXTM3U
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="English", \
    DEFAULT=YES,AUTOSELECT=YES,LANGUAGE="en", \
    URI="main/english-audio.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="Deutsche", \
    DEFAULT=NO,AUTOSELECT=YES,LANGUAGE="de", \
    URI="main/german-audio.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="Commentary", \
    DEFAULT=NO,AUTOSELECT=NO,URI="commentary/audio-only.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=1280000,CODECS="...",AUDIO="aac"
low/video-only.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=2560000,CODECS="...",AUDIO="aac"
mid/video-only.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=7680000,CODECS="...",AUDIO="aac"
hi/video-only.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=65000,CODECS="mp4a.40.5",AUDIO="aac"
main/english-audio.m3u8
```

8.8. Variant Playlist with Alternative video

In this example, the PROGRAM-ID attributes have been left out. The
CODECS attributes have been condensed for space. A '\' is used to
indicate that the tag continues on the following line with whitespace
removed:

```
#EXTM3U
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="low",NAME="Main", \
    DEFAULT=YES,URI="low/main/audio-video.m3u8"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="low",NAME="Centerfield", \
    DEFAULT=NO,URI="low/centerfield/audio-video.m3u8"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="low",NAME="Dugout", \
    DEFAULT=NO,URI="low/dugout/audio-video.m3u8"

#EXT-X-STREAM-INF:BANDWIDTH=1280000,CODECS="...",VIDEO="low"
low/main/audio-video.m3u8

#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="mid",NAME="Main", \
    DEFAULT=YES,URI="mid/main/audio-video.m3u8"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="mid",NAME="Centerfield", \
    DEFAULT=NO,URI="mid/centerfield/audio-video.m3u8"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="mid",NAME="Dugout", \
    DEFAULT=NO,URI="mid/dugout/audio-video.m3u8"

#EXT-X-STREAM-INF:BANDWIDTH=2560000,CODECS="...",VIDEO="mid"
mid/main/audio-video.m3u8

#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="hi",NAME="Main", \
    DEFAULT=YES,URI="hi/main/audio-video.m3u8"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="hi",NAME="Centerfield", \
    DEFAULT=NO,URI="hi/centerfield/audio-video.m3u8"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="hi",NAME="Dugout", \
    DEFAULT=NO,URI="hi/dugout/audio-video.m3u8"

#EXT-X-STREAM-INF:BANDWIDTH=7680000,CODECS="...",VIDEO="hi"
hi/main/audio-video.m3u8

#EXT-X-STREAM-INF:BANDWIDTH=65000,CODECS="mp4a.40.5"
main/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 in iOS 3.0 and later and QuickTime Player
in Mac OS X version 10.6 and later.

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 segments received from a server
that non-compliant segments 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 6265 [RFC6265]. See also the Security Considerations section of RFC 6265, 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.

   [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.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246, August 2008.

   [RFC5646] Phillips, A. and M. Davis, "Tags for Identifying
              Languages", BCP 47, RFC 5646, September 2009.

   [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70,
              RFC 5652, September 2009.

   [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265,
              April 2011.

   [RFC6381] Gellens, R., Singer, D., and P. Frojdh, "The 'Codecs' and
              'Profiles' Parameters for "Bucket" Media Types", RFC 6381,
              August 2011.

   [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>.

Authors' Addresses

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

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

    William May, Jr.
    Apple Inc.
    Cupertino, California
    United States

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