Protocol Extension for Low-Latency HLS (Preliminary Specification)

This is the preliminary specification and implementation guide for adding Low-Latency HLS to your streams.

On This Page

- Overview
- See Also

Overview

The HTTP Live Streaming (HLS) protocol delivers live and on-demand content streams to global-scale audiences. HLS has historically favored stream reliability over latency. Low-Latency HLS extends the protocol to enable low-latency video streaming while maintaining the same degree of scalability. The new low-latency mode lowers video latencies over public networks into the range of standard television broadcasts.

Backend production tools and content delivery systems must implement new rules to enable low-latency stream playback. This document outlines Low-Latency HLS functionality and the requirements for producing and delivering Low-Latency HLS streams. It also provides examples of Low-Latency HLS playlists.

New Functionality

Low-Latency HLS is an extension of the existing HLS protocol documented in RFC 8216 and draft-pantos-hls-rfc8216bis. As detailed in the sections that follow, Low-Latency HLS offers new functionality in these areas:

- Generation of Partial Segments
- Playlist Delta Updates
- Blocking of playlist reload
- Preload hints and blocking of Media downloads
- Rendition Reports

The Server Configuration Profile Requirements section describes the server configuration profile that distribution systems must support in order to engage the low-latency playback mode. A server indicates compatibility with the low-latency mode with a new Media Playlist tag, `EXT-X-SERVER-CONTROL`.

Generation of Partial Media Segments

Low-Latency HLS provides a parallel channel for distributing media at the live edge of the Media Playlist, where the media is divided into a larger number of smaller files, such as CMAF Chunks. These smaller files are called HLS Partial Segments. Because each Partial Segment has a short duration, it can be packaged, published, and added to the Media Playlist much earlier than its Parent Segment. While regular Media Segments might be 6s each, an example Partial Segment might be only 200ms. A first Partial Segment might be published only 200ms after the previous segment has been published, followed by 29 of its peers, followed at last by a regular 6s Media Segment. This Media Segment contains the same media as the concatenation of its 30 Partial Segments. In order to reduce playlist bloat, Partial Segments are removed from the Media Playlist once they are greater (older) than 3 target durations from the live edge.

Deleted: automatically

You add Partial Segments to the Media Playlist using the new `EXT-X-PART` tag. You may use other Media Segment Tags (such as `EXT-X-DISCONTINUITY`) only at segment boundaries, not between Partial Segments.

A Partial Segment must be in one of the Supported Media Segment Formats described by the HLS specification (sections 3.1 through 3.5).

Playlist Delta Updates

Playlists are transferred more frequently with Low-Latency HLS. Clients can request and servers can provide Playlist Delta Updates to reduce the transfer cost. These updates replace a considerable portion of the Playlist that the client already has with the new `EXT-X-SKIP` tag.

Blocking of Playlist Reload

To support efficient client notification of new Media Segments and Partial Segments, Low-Latency HLS introduces the ability to block a playlist reload request. When a client issues an HTTP GET to request a Media Playlist update, it can add query parameters to specify that it wants the playlist response to include a future segment. It is the responsibility of the server to hold onto the request (block) until a version of the playlist that contains that segment is available. ~~(The client can also ask the server to push the indicated segment to the client along with the playlist response.)~~ Blocking playlist reload eliminates polling. ~~(Server push eliminates request round trips.)~~

Deleted: contain

Formatted: Strikethrough

Formatted: Strikethrough

Preload Hints and Blocking of Media Downloads

Eliminating unnecessary round trips is critical when delivering low-latency streams at global scale. Servers use a new tag, EXT-X-PRELOAD-HINT, to inform clients of upcoming Partial Segments and Media Initialization Sections. A client can issue a GET request for a hinted resource in advance; the server responds to the request as soon as the media becomes available.

Deleted: When a

Deleted: s

Deleted: a hinted resource

Deleted: ,

Deleted: must hold onto

Deleted: until

Deleted: the entire response can be delivered to the client at link speed. This enables the client to perform accurate Adaptive Bit Rate (ABR) measurements.

Rendition Reports

When playing at low latency, the client must be able to switch renditions with a minimum number of round trips in order to perform bit-rate adaptation. To support this, the server must add Rendition Reports on the other renditions in the Master Playlist to each Media Playlist. A Rendition Report is carried in the `EXT-X-RENDITION-REPORT` tag and provides information such as the last Media Sequence Number and Part currently in the Media Playlist of that rendition.

Playlist Request Query Parameters for Low-Latency HLS

Clients must ensure that all query parameters of the URL of a GET request for a low-latency playlist appear in UTF-8 order within the URL. This increases CDN cache utilization. You can use the following new query parameters with Low-Latency HLS:

- `_HLS_msn=<N>`: Indicates that the server must hold the request until a playlist contains a Media Segment with Media Sequence Number of N or later. The server must deliver the entire playlist, even if the requested Media Segment is not the last in the playlist, or in fact if it is no longer in the playlist.
- `_HLS_part=<M>`: Use in combination with `_HLS_msn` to indicate that the server must hold the request until a playlist contains Partial Segment M of Media Sequence Number of N or later. The first Partial Segment of a segment is `_HLS_part=0`, the second is `_HLS_part=1`, and so on. The `_HLS_part` parameter requires an `_HLS_msn` parameter.

If the client asks for a Partial Segment number greater than the final part of a segment, the server must deliver a playlist containing the first Partial Segment of the following segment.

If a client supplies an `_HLS_msn` parameter greater than the Media Sequence Number of the last segment in the current Playlist plus 2, or if it supplies an `_HLS_part` parameter that exceeds the last Partial Segment in the current playlist by the Advance Part Limit, then the server should immediately return 400.

The Advance Part Limit is 3 divided by the PART-TARGET if the PART-TARGET is less than 1 second, or 3 otherwise.

`_HLS_msn` and `_HLS_part` do not trigger blocking if the playlist contains an `EXT-X-ENDLIST`tag.

A 3x target duration timeout is recommended for blocking requests, after which the server should return 503.

- (~~`_HLS_push=<0/1>`: Indicates whether the server must push the awaited Partial Segment along with the playlist response. `1` means push, `0` means don't push. The absence of an `_HLS_push` parameter is equivalent to `_HLS_push=0`.~~)
- `_HLS_skip=YES`: Requests a Playlist Delta Update, in which the earlier portion of the playlist is replaced with an `EXT-X-SKIP` tag. The server must ignore this parameter if the playlist contains an `EXT-X-ENDLIST` tag.

New Media Playlist Tags for Low-Latency HLS

Use the following playlist tags when you implement Low-Latency HLS.

EXT-X-SERVER-CONTROL:<attribute-list>

`EXT-X-SERVER-CONTROL` allows the server to indicate support for features such as Blocking Playlist Reload and Playlist Delta Updates. This tag is required when you implement Low-Latency HLS.

All low-latency Media Playlists must carry this tag with the same attributes and values. The attribute list can consist of the following attributes:

- `CAN-BLOCK-RELOAD=YES`: Indicates that the server supports processing the `_HLS_msn` and `_HLS_part` parameters, blocking requests for hinted resources, and the publishing of Rendition Reports. It is mandatory for Low-Latency HLS.
- `CAN-SKIP-UNTIL=<seconds>`: (optional) Indicates that the server can produce Playlist Delta Updates in response to the `_HLS_skip=YES` parameter. Its value is the Skip Boundary, as a floating-point number of seconds. Segments and their associated tags that are further from the live edge than the Skip Boundary can be replaced by an `EXT-X-SKIP` tag in a Playlist Delta Update. The Skip Boundary must be at least 6 times the `EXT-X-TARGETDURATION`. You can use this attribute in low-latency and regular HLS playlists.
- `HOLD-BACK=<seconds>`: (optional) Indicates the server-recommended minimum distance from the live edge at which clients should begin to play or to which they should seek when NOT playing in low-latency mode (that is,

Formatted: Strikethrough

Deleted: ,

Deleted: , and `_HLS_push`

conventional HLS). Its value is a floating-point number of seconds and must be at least 3 times the EXT-X-TARGETDURATION.

- PART-HOLD-BACK=<seconds>: Indicates the server-recommended hold-back time when playing in low-latency mode. It is mandatory if the Playlist contains EXT-X-PART tags. Its value is a floating-point number of seconds and must be at least PART-TARGET. The recommended time is at least 3 x PART-TARGET.

Deleted: PART-HOLD-BACK

EXT-X-PART-INF:<attribute-list>

EXT-X-PART-INF provides information about HLS Partial Segments in the playlist. It is required if a playlist contains one or more EXT-X-PART tags. The attribute list consists of the following attribute:

- PART-TARGET=<s>: (mandatory) Indicates the part target duration in floating-point seconds and is the maximum duration of any Partial Segment.

All Partial Segments except the last part of a segment must have a duration of at least 85% of PART-TARGET. The server must add a new Partial Segment to the Playlist every part target duration.

EXT-X-PART:<attribute-list>

EXT-X-PART identifies a Partial Segment in the playlist.

The set of EXT-X-PART tags between EXTINF tags must represent the same set of media as the Media Segment of the following EXTINF tag. The Media Segment containing the same media as a set of Partial Segments is called the Parent Segment of those Partial Segments.

A Partial Segment must be completely available for download at the full speed of the link to the client at the time it is added to the playlist.

All Media Segment tags except for EXT-X-DATERANGE, EXT-X-BYTERANGE, and EXT-X-GAP that are applied to a Parent Segment must appear before the first EXT-X-PART tag of the Parent Segment. These tags include EXT-X-MAP, EXT-X-DISCONTINUITY, and EXT-X-KEY.

Deleted: with the exception of

Deleted:

Remove EXT-X-PART tags from the playlist after they are greater than 3 target durations from the end of the playlist. Partial Segments are primarily useful for navigating close to the live edge, after which their presence does not justify the increase in the playlist size and the responsibility of retaining the parallel Partial Segment stream on the server. The attribute list can consist of the following attributes:

- DURATION=<s>: (mandatory) Indicates the duration of the Partial Segment in floating-point seconds.
- URI=<url>: (mandatory) Indicates the URI for the Partial Segment.
- INDEPENDENT=YES: Indicates that the Partial Segment contains an independent frame. It is mandatory for such Partial Segments, unless every Partial Segment contains an independent frame.
- BYTERANGE=<n>[@<o>]: Indicates that the Partial Segment is a subrange of the resource specified by the URI attribute. It is mandatory for such Partial Segments.
- GAP=YES: Indicates that the Partial Segment is not available. It is mandatory for such Partial Segments. The rules that apply to a Media Segment with an EXT-X-GAP tag applied to it also apply to Partial Segments with a GAP=YES attribute. A Parent Segment must have an EXT-X-GAP tag applied to it if one or more of its Partial Segments have a GAP=YES attribute. The EXT-X-GAP tag should appear immediately after the first EXT-X-PART tag in the Parent Segment with a GAP=YES attribute.

EXT-X-PRELOAD-HINT:<attribute-list>

The EXT-X-PRELOAD-HINT tag is used to hint that a resource or a byte range of a resource will be needed to play back an upcoming part of the presentation. The hinted resource must be available for request when the EXT-X-PRELOAD-HINT tag is added to the Playlist.

The server must not transmit any range of a Partial Segment from a hinted resource until an EXT-X-PART tag specifying that range is added to the Playlist. This enables the client to perform accurate Adaptive Bit Rate (ABR) measurements.

The server should respond with 404 if it receives a request for a resource which cannot be found by the server and is not specified by an EXT-X-PRELOAD-HINT tag in an active (i.e. live) Media Playlist.

Deleted: The server must not transmit the hinted resource until the entire response can be sent at link speed.

The attribute list can consist of the following attributes:

- TYPE=<hint-type>: (mandatory) Specifies the type of the hinted resource. If hint-type is PART, the resource is an upcoming Partial Segment. If hint-type is MAP, the resource is an upcoming Media Initialization Section.
- URI=<uri>: (mandatory) Specifies the hinted resource. It must match the URI string that will be subsequently added to the playlist as a non-hinted resource (for example as part of an EXT-X-PART tag). The URI may be relative to the URI of the Playlist or it may be absolute. The hostname may or may not match.

Formatted: Font:(Default) Helvetica Neue, Condensed by 0.25 pt

- BYTERANGE-START=<n>: Indicates that the requested resource begins at a particular byte offset from the beginning of the resource identified by the URI attribute. Its absence implies a value of 0.
- BYTERANGE-LENGTH=<n>: Indicates the length of the requested resource, which may be less than the length of the resource identified by the URI attribute. Its absence indicates that the specified range extends from the start offset to the end of the resource identified by the URI attribute.

Note that when a hinted Partial Segment eventually appears in the Playlist as an EXT-X-PART tag, it may have a different Discontinuity Sequence Number, a different Media Initialization Section, and/or a different encryption configuration than the previous Partial Segment. In other words, it can be preceded by an EXTINF tag indicating the end of the previous Parent Segment and an EXT-X-DISCONTINUITY, EXT-X-MAP, and/or EXT-X-KEY tag.

Formatted: None, Space Before: 0 pt

If the Playlist contains EXT-X-PART tags and does not contain an EXT-X-ENDLIST tag, the Playlist must contain an EXT-X-PRELOAD-HINT tag with a TYPE=PART attribute to hint the URI of the next EXT-X-PART tag that is expected to be added to the Playlist (and its byte range, if applicable). Upon reading such a Playlist a client with sufficient space in its download pipeline that is not already loading the hinted resource should issue a GET request for it. This will typically be issued at the same time as its GET request for the next Playlist update.

If the hinted Partial Segment specified by the TYPE=PART EXT-X-PRELOAD-HINT tag has a different Media Initialization Section than the last Partial Segment in the Playlist, the Playlist must also contain an EXT-X-PRELOAD-HINT tag with a TYPE=MAP attribute that identifies the Media Initialization Section of the hinted Partial Segment. A client with sufficient space in its download pipeline that has not already cached the hinted Media Initialization Section should issue a GET request for it.

Servers should not add more than one EXT-X-PRELOAD-HINT tag with the same TYPE attribute to a Playlist. Clients should ignore all but the first EXT-X-PRELOAD-HINT tag with a particular TYPE attribute in a Playlist. Clients must ignore EXT-X-PRELOAD-HINT tags with unrecognized TYPE attributes.

A server may choose not to publish a previously-hinted Partial Segment if planned segmentation changes, such as the case of early return from an ad. It should respond to client requests for that Partial Segment with a 404.

Formatted: None, Space Before: 0 pt

A client should abandon the download of a hinted resource if it does not appear in a subsequent Playlist update, either as in EXT-X-PRELOAD-HINT tag or as part of another tag such as EXT-X-PART. It should ignore the result code of such a GET request.

If a Partial Segment is created as a sub-range of a larger resource and its length is not known at the time that its hint is added to the Playlist, the BYTERANGE-LENGTH attribute should be omitted. The BYTERANGE-OFFSET of its hint should indicate its starting offset into the larger resource. This will signal the client to issue a download request from the start of the hinted Partial Segment to the end of the resource.

A client should recognize when a Partial Segment indicated by an EXT-X-PART tag is a sub-range of a hint download and obtain the Partial Segment from the hint download. Clients must recognize contiguous ranges between existing Partial Segments and Partial Segment hints and avoid duplicate downloads.

A server should not hint a range that it does not expect to be downloaded by clients in the near term.

Formatted: Font:(Asian) +Theme Body Asian (M S 明朝), 13 pt, Condensed by 0.25 pt

EXT-X-RENDITION-REPORT:<attribute-list>

EXT-X-RENDITION-REPORT carries information about an associated rendition that is as up-to-date as the playlist that contains it.

The server must add one EXT-RENDITION-REPORT tag for each Media Playlist (Rendition) in the Master Playlist, except for the Media Playlist to which the EXT-X-RENDITION-REPORT tag is being added and Playlists which contain the EXT-X-I-FRAMES-ONLY tag. The attribute list can consist of the following attributes:

- URI=<uri>: (mandatory) Indicates the Media Playlist described in the report. It should be the same string as the report query parameter.
- LAST-MSN=<N>: (mandatory) Indicates the Media Sequence Number of the last segment currently in the specified rendition. If the rendition contains Partial Segments then this value is the Media Sequence Number of the segment containing the last Partial Segment.
- LAST-PART=<M>: Indicates the last part of the segment specified by the media sequence number currently in the specified rendition. It is mandatory if the associated playlist contains EXT-X-PART tags.

A server may omit adding an attribute to an EXT-X-RENDITION-REPORT tag — even a mandatory attribute — if its value is the same as that of the Rendition Report of the Media Playlist to which the `EXT-X-RENDITION-REPORT` tag is being added. This will reduce the size of the Rendition Report.

Currently the prototype tools also generate `LAST-I-MSN` and `LAST-I-PART` parameters to indicate the MSN and part of the last independent part. But these may not make it into the specification because they are currently unused.

EXT-X-SKIP:<attribute-list>

When a server issues a Playlist Delta Update, it replaces Media Segments earlier than the Skip Boundary and their associated tags with an `EXT-X-SKIP` tag.

The `EXT-X-SKIP` tag replaces segment URI lines and all of the following tags that are applied to those segments: `EXTINF`, `EXT-X-BYTERANGE`, `EXT-X-DISCONTINUITY`, `EXT-X-KEY`, `EXT-X-MAP`, `EXT-X-PROGRAM-DATE-TIME`, `EXT-X-GAP`, and `EXT-X-BITRATE`. All other tags must remain in the Playlist Delta Update. The attribute list can consist of the following attributes:

- `SKIPPED-SEGMENTS=<N>`: (mandatory) Indicates how many Media Segments were replaced by the `EXT-X-SKIP` tag, along with their associated tags.

If a client receives a playlist containing an `EXT-X-SKIP` tag and discovers that it does not already have all of the information that was skipped, it must obtain a complete copy of the playlist by reissuing its playlist request without the `_HLS_skip=YES` parameter.

A playlist containing an `EXT-X-SKIP` tag must have an `EXT-X-VERSION` tag with a value of 9 or higher.

Server Configuration Profile Requirements

To support timely delivery of media, Low-Latency HLS requires certain transport features above and beyond what is necessary for regular HLS. Clients must verify that these features are in place before engaging low-latency mode. Because the Low-Latency HLS syntax is backward-compatible with existing HLS, clients will fall back to regular-latency HLS playback if they discover that the server does not support an aspect of the required configuration.

You must serve Low-Latency HLS streams via HTTP/2 because efficient delivery requires HTTP/2 features such as multistream control and Ping requests. Servers must support H2 priority control (dependencies and weights). TCP is recommended, and there is no commitment to support QUIC in the first release. Each server must offer the entire set of tiers in the master playlist. This allows rapid tier switching without connection reestablishment. Servers must support HTTP

Deleted: they

Deleted: several

Deleted: :

Deleted: , H2 Push,

Deleted: H2

Deleted:

Range requests if Media Playlists contain BYTERANGE, BYTERANGE-START or BYTERANGE-LENGTH attributes.

TCP implementations must support Selective Acknowledgment (SACK) across the entire route from client to server. You should also set Explicit Congestion Notification (ECN) during congestion, and use TCP timestamps, TAIL LOSS probe, and TCP RACK. These additions improve the performance of TCP loss recovery. See RFC 2018, RFC 3168, RFC 7323, and IETF draft-ietf-tcpm-rack for more information about these TCP options.

Playlist requests must be idempotent. Servers should support TLS 1.3 or higher to reduce connection time. Servers should also support TLS 1.3 0-RTT connections for Media Playlists and Media Segments.

Playlists (but not segments) must be in GZIP format. This speeds up Media Playlist reload and rendition switching.

CDNs and proxy caches must recognize client playlist requests and blocking media requests that match an existing request that is currently outstanding to the origin, and must hold the duplicate requests until the origin responds to the existing request. This is more critical in Low-Latency HLS due to its requirements for an active origin. CDNs have different names for this feature; for example, Apache Traffic Server calls it reader-while-writer.

Low-Latency HLS allows longer caching of playlists without detriment to clients. It is recommended that you cache successful playlist requests and responses with Low-Latency HLS query parameters for 6 target durations, and cache unsuccessful requests/responses (such as 404s) for 4 target durations. (It is recommended that you cache successful responses to playlist requests that don't contain the `_HLS_msn` query parameter for 0.5 target duration and unsuccessful responses for 1 target duration.) It is recommended that unsuccessful responses to blocking media requests be cached for one target duration.

Origins should use cache-control headers to indicate the desired cache lifetime.

Different renditions must update in sync, to within 1 part-target duration.

HTTP caches used for delivery of Low-Latency HLS must set the Age HTTP Response header.

Revisions to Existing Media Authoring Rules for Low-Latency HLS Streams

Media Playlists must have `EXT-PROGRAM-DATE-TIME` tags. This allows more precise mapping between segments across renditions. (Note that real-time clocks are NOT required to be synchronized between client and server). The recommended 6s target duration still applies. The recommended GOP size is 1-2s. Smaller GOPs support faster switching between renditions.

Example: Low-Latency HLS Playlist

```
#EXTM3U
# This playlist is a response to: GET
https://example.com/2M/waitForMSN.php?_HLS_msn=273&_HLS_part=2
#EXT-X-TARGETDURATION:4
#EXT-X-VERSION:6
#EXT-X-SERVER-CONTROL:CAN-BLOCK-RELOAD=YES,PART-HOLD-BACK=1.0,CAN-
SKIP-UNTIL=12.0
#EXT-X-PART-INF:PART-TARGET=0.33334
#EXT-X-MEDIA-SEQUENCE:266
#EXT-X-PROGRAM-DATE-TIME:2019-02-14T02:13:36.106Z
#EXT-X-MAP:URI="init.mp4"
#EXTINF:4.00008,
fileSequence266.mp4
#EXTINF:4.00008,
fileSequence267.mp4
#EXTINF:4.00008,
fileSequence268.mp4
#EXTINF:4.00008,
fileSequence269.mp4
#EXTINF:4.00008,
fileSequence270.mp4
#EXT-X-PART:DURATION=0.33334,URI="filePart271.0.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.1.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.2.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.3.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.4.mp4",INDEPENDENT=YES
#EXT-X-PART:DURATION=0.33334,URI="filePart271.5.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.6.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.7.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.8.mp4",INDEPENDENT=YES
#EXT-X-PART:DURATION=0.33334,URI="filePart271.9.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.10.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.11.mp4"
#EXTINF:4.00008,
fileSequence271.mp4
#EXT-X-PROGRAM-DATE-TIME:2019-02-14T02:14:00.106Z
#EXT-X-PART:DURATION=0.33334,URI="filePart272.a.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.b.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.c.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.d.mp4"
```

Deleted: &_HLS_report=../1M/waitForMSN.php &_HLS_report=../4M/waitForMSN.php

Deleted: 3

Deleted:

Deleted: ts

Deleted:

Deleted: ts

Deleted:

Deleted: ts

Deleted:

Deleted: ts

Deleted:

Deleted: ts

Deleted:

Deleted: ts

Deleted: .ts

Deleted: .ts

Deleted: .ts

Deleted: .ts

Deleted: .ts

Deleted: .ts

Deleted: .ts

Deleted: .ts

Deleted: .ts

Deleted: .ts

Deleted: Part

Deleted: .ts

Deleted: .ts

Deleted: 13

Deleted: 60

Deleted: .ts

Deleted: .ts

Deleted: .ts

Deleted: .ts

```
#EXT-X-PART:DURATION=0.33334,URI="filePart272.e.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.f.mp4",INDEPENDENT=YES
#EXT-X-PART:DURATION=0.33334,URI="filePart272.g.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.h.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.i.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.j.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.k.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.l.mp4"
#EXTINF:4.00008,
fileSequence272.mp4
#EXT-X-PART:DURATION=0.33334,URI="filePart273.0.mp4",INDEPENDENT=YES
#EXT-X-PART:DURATION=0.33334,URI="filePart273.1.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart273.2.mp4"
#EXT-X-PRELOAD-HINT:TYPE=PART,URI="filePart273.3.mp4"

#EXT-X-RENDITION-REPORT:URI="../1M/waitForMSN.php",LAST-
MSN=273,LAST-PART=2
#EXT-X-RENDITION-REPORT:URI="../4M/waitForMSN.php",LAST-
MSN=273,LAST-PART=1
```

Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Formatted: Font color: R,G,B (209,47,27)

Example: Playlist Delta Update

```
#EXTM3U
# Following the example above, this playlist is a response to: GET
https://example.com/2M/waitForMSN.php?_HLS_msn=273&_HLS_part=3
&_HLS_skip=YES
#EXT-X-TARGETDURATION:4
#EXT-X-VERSION:9
#EXT-X-SERVER-CONTROL:CAN-BLOCK-RELOAD=YES,PART-HOLD-BACK=1.0,CAN-
SKIP-UNTIL=12.0
#EXT-X-PART-INF:PART-TARGET=0.33334
#EXT-X-MEDIA-SEQUENCE:266
#EXT-X-SKIP:SKIPPED-SEGMENTS=3
#EXTINF:4.00008,
fileSequence269.mp4
#EXTINF:4.00008,
fileSequence270.mp4
#EXT-X-PART:DURATION=0.33334,URI="filePart271.0.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.1.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.2.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.3.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.4.mp4",INDEPENDENT=YES
#EXT-X-PART:DURATION=0.33334,URI="filePart271.5.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.6.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.7.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.8.mp4",INDEPENDENT=YES
```

Deleted: &_HLS_report=../1M/waitForMSN.php &_HLS_report=../4M/waitForMSN.php
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts

```
#EXT-X-PART:DURATION=0.33334,URI="filePart271.9.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.10.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart271.11.mp4"
#EXTINF:4.00008,
fileSequence271.mp4
#EXT-X-PROGRAM-DATE-TIME:2019-02-14T02:14:00.106Z
#EXT-X-PART:DURATION=0.33334,URI="filePart272.a.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.b.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.c.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.d.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.e.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.f.mp4",INDEPENDENT=YES
#EXT-X-PART:DURATION=0.33334,URI="filePart272.g.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.h.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.i.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.j.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.k.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart272.l.mp4"
#EXTINF:4.00008,
fileSequence272.mp4
#EXT-X-PART:DURATION=0.33334,URI="filePart273.0.mp4",INDEPENDENT=YES
#EXT-X-PART:DURATION=0.33334,URI="filePart273.1.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart273.2.mp4"
#EXT-X-PART:DURATION=0.33334,URI="filePart273.3.mp4"
#EXT-X-PRELOAD-HINT:TYPE=PART,URI="filePart273.4.mp4"

#EXT-X-RENDITION-REPORT:URI="../1M/waitForMSN.php",LAST-
MSN=273,LAST-PART=3
#EXT-X-RENDITION-REPORT:URI="../4M/waitForMSN.php",LAST-
MSN=273,LAST-PART=3
```

Example: Byterange-addressed Parts

```
# In these examples only the end of the Playlist is shown.
# This is Playlist update 1
#EXTINF:4.08,
fs270.mp4
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=20000@0
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=23000@20000
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=18000@43000
#EXT-X-PRELOAD-HINT:TYPE=PART,URI="fs271.mp4",BYTERANGE-START=61000

# This is Playlist update 2
#EXTINF:4.08,
fs270.mp4
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=20000@0
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=23000@20000
```

Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: 13
Deleted: 60
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts
Deleted: .ts

```
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=18000@43000
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=19000@61000
#EXTINF:4.08,
fs271.mp4
#EXT-X-PRELOAD-HINT:TYPE=PART,URI="fs272.mp4",BYTERANGE-START=0

# This is Playlist update 3
#EXTINF:4.08,
fs270.mp4
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=20000@0
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=23000@20000
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=18000@43000
#EXT-X-PART:DURATION=1.02,URI="fs271.mp4",BYTERANGE=19000@61000
#EXTINF:4.08,
fs271.mp4
#EXT-X-PART:DURATION=1.02,URI="fs272.mp4",BYTERANGE=21000@0
#EXT-X-PRELOAD-HINT:TYPE=PART,URI="fs272.mp4",BYTERANGE-START=21000
```

Appendix A: CDN Tune-In

Players and other clients of Low-Latency HLS should expect delivery of low-latency streams through CDNs and other HTTP caches. To correctly implement the server-recommended playback distance from the live feed, `PART-HOLD-BACK`, the client must first obtain a reasonably up-to-date version of the Media Playlist.

There are various approaches that a client may take to obtain a recent version of a Media Playlist. The following algorithm typically rquires two GET requests to obtain a Playlist that's within one part target duration of the current Playlist:

1. Send a request for the Media Playlist that doesn't include an `_HLS_msn` or `_HLS_part`parameter.
2. Record the first Playlist response, including its received time and Age header. If there's no Age header in the first Playlist response, consider the Playlist to be up to date.
3. If there's an Age header in the first Playlist response, set the `goalDuration` to match the Age value. Add 1 second to the `goalDuration` value if the part target duration is less than 1.0.

While the Age value is greater than or equal to the floor of the part target duration:

1. Set `currentGoal` to be the `goalDuration` plus the amount of time since the first Playlist response.
2. If the current version of the Playlist has at least `currentGoal` more media in it than the first Playlist, consider the current Playlist to be up to date.

3. Use the target duration and the part target duration to estimate how many more segments and parts the server will add to the Playlist to contribute at least `currentGoal` more media to it.
4. Request the Media Playlist again, using the `_HLS_msn` and `_HLS_part` parameters to obtain the Playlist that has the estimated additional duration of media since the first Playlist.
5. Update the current Playlist and the Age value from the Playlist response.

Revision History

The following table describes the changes to the Protocol Extension for Low-Latency HLS (Preliminary Specification).

Date	**Notes**
2019/12/22	Replaced the use of HTTP/2 Push with EXT-X-PRELOAD-HINT. Switched examples to mp4. Added byte range example.
2019/08/22	Removed the `_HLS_report` parameter and made Rendition Reports mandatory. Clarified the `LAST-MSN` definition.
2019/07/25	Added the CDN Tune-in appendix.
2019/07/10	Updated MSN and part validation rules. Made server response recommended instead of mandatory. Added the requirement of the AGE header when using HTTP proxy caches.
2019/06/03	New document describing the low-latency HTTP Live Streaming.