Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 8, 2019

O. Finkelman
Qwilt
A. Begen
Networked Media
November 4, 2018

HTTP Redirects in HTTP Adaptive Streaming
draft-finkelman-httpbis-has-redirecting-00

Abstract

This document motivates the need for clarifications of client behavior in cases of HTTP redirect to a content delivery network (CDN) when the redirected object contains relative references. This document focuses on HTTP Adaptive Streaming (HAS) use cases, but it might be possible to generalize to other use cases. The goal of this document is to present the current status of things and to explore potential solutions.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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 https://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 May 8, 2019.

Copyright Notice

Table of Contents

1. Introduction

HTTP redirect, commonly using the "302 Found" response code, is widely used in Content Delivery Networks (CDN) for HTTP Adaptive Streaming (HAS), specifically for HTTP Live Streaming (HLS) [RFC8216] and Dynamic Adaptive Streaming over HTTP (DASH) [MPEG-DASH]. Using HTTP redirect rather than delegation by CNAME is a more powerful and flexible solution.

At a high level, the required behavior from a HAS client is that after getting an HTTP redirect to a playlist address that takes the client to a CDN cache, if the playlist contains relative references to the media segments, the client should be 'sticky' and request the subsequent media segments directly from the cache without the need for repeated redirects, which naturally would add undesirable latency. Unfortunately, in some cases, common misinterpretations of the RFC lead to wrong behavior of clients, while in other cases, this required behavior contradicts the HTTP RFC. Thus, we are facing

issues with streaming clients that behave differently under different use cases of HTTP redirects when relative references are in use within the playlist.

The Streaming Video Alliance (SVA) and DASH Industry Forum (DASH-IF) are attempting to resolve these issues and create clear guidelines describing how video players, browsers and streaming applications should behave in such HTTP redirect cases.

A discussion of this use case can also be found in Section 2.2.1 of [RFC6983].

The following sequence explains this step by step:

1. The client gets a URI to the playlist file from the origin website (content provider). This URI may point directly at a CDN address or be CNAMEd to a CDN node.

2. The client requests the playlist file from the CDN.

3. The CDN responds with an HTTP redirect, or a series of redirects that eventually takes the client to the final cache.

4. The client requests the playlist from this final cache.

5. The client parses the relative URIs to media segments from the playlist.

6. The client uses the final cache address to construct the absolute URIs for the media segments that point to the final cache.

7. The client requests the media segments from the final cache.

The redirection should still be temporary in the sense that if a client does a refresh, or repeatedly fails to get the playlist or the media files from the cache address, it can go back to the original playlist URI and request it again.

2. Glossary of Terms

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Current State of Things

Currently, there are inconsistencies in the behaviors of HAS clients when constructing the URI of a relative referenced object, read from another resource, which was redirected. In some cases the client constructs these URI relative to the redirected address, while in

other cases the client constructs it relative to the original URI of the playlist before the redirection.

The result of the latter behavior is that the client returns to the original host for each and every request, and then gets redirected again to the cache address. This has a significant negative impact on the the user experience (due to increased risk of slower media segment downloads). Further, it causes redundant work in the redirecting CDN host. The reasons for this are, in some cases, misinterpretation of the standard and, in other cases, contradictions between this requirement and the existing standard.

4. Objectives

Our objectives are to document a clear understanding when a client SHOULD compute the absolute URI for relative reference taken from a playlist using the actual retrieval URI of the playlist, and have a specification that leads to a consistent behavior that meets the HAS requirement, regardless of how the retrieval URI of the playlist was resolved.

5. Potential Solutions

The below are high-level initial thoughts in order to get the feedback from the HTTPbis working group. As we see it, the alternatives could be one of the following, depending on the understanding of the current state of things:

1. Have a single consistent interpretation of the existing RFCs

showing in which cases the behavior should be as required above.

2. Draft an extension to Section 5 of [RFC3986] to handle the required cases.

3. Create a new HTTP 3xx response code that will be specified to serve the required use cases.

4. Handle it outside the HTTP standard, i.e., in the SVA and DASH-IF.

A solution should take under consideration the more advanced use cases of:

1. A chain of redirects to get to the final cache.

2. Redirect back: What happens if there is a need to redirect the client back to the original URI? How does the client behave

after such redirect of a specific media object? Is there any difference between this case and another forward redirection?

3. An absolute URI appearing in the middle of a playlist, between relative references (for example, for an advertisement segment) should not break the sequence of things, it should be treated as an absolute URI and the subsequent segments that are relative references should be requested from the cache again.

4. Failover: If the cache does not answer, the client should go back and request the playlist from the original URI, and not request the relative reference object directly as it does not know how to construct an alternative URI for it.

5. The client MAY, on its own consideration, re-request the playlist again from the original URI, and may be redirected again to the same cache, or other cache. In that sense the redirect is temporary.

6. The solution should handle both the use case in which the encapsulating entity was redirected and it includes relative references to the media segment, as in HLS (Section 5.1.3 of

[RFC3986]), and the use case that encapsulating entity includes the base URL embedded in it, as in DASH (Section 5.1.1 of [RFC3986]), which may lead to the understanding that DASH services should avoid using an absolute URI as base URL in the Media Presentation Description (MPD), if they wish HTTP redirect to CDN to work properly.

6. Examples and Interpretation of the Current RFCs

This is our interpretation of the current state of things using HLS manifest examples, in each example the manifest URI is resolved differently and we observe the implications on the resolution of the relative references contained within the manifest file.

In the first example the master manifest URI is given as an absolute URI, while in the second example the master manifest is given itself as a relative reference from the encapsulation that provided it, both lead to the required behavior.

In the third example, similar to the second, the manifest is given as a relative reference but the request for the manifest is redirected. This leads to the undesired behavior of the client going back to the original location for every request, requiring redirect for every media chunk.

6.1. Example 1: Absolute URI to Manifest

Let us take a HLS example and assume the player is passed an absolute URI for the HLS manifest (e.g., a user has entered the manifest address directly into the player):

https://video.cdn.example.com/vod/movie/master.m3u8

GET /vod/movie/master.m3u8

Host: video.cdn.example.com

HTTP/1.1 302 Found

Location: https://edge1.dcdn.example.net/vod/movie/master.m3u8

The client then requests the HLS manifest from the edge cache.

```
GET /vod/movie/master.m3u8

Host: edge1.dcdn.example.net

HTTP/1.1 200 OK
```

The client reads the per-bitrate manifest reference from the master manifest.

```
#EXT-X-STREAM-INF:BANDWIDTH=2121091,RESOLUTION=640x360,CODECS="mp4
a.40.2,avc1.4d001f"

1800K/1800_complete.m3u8
```

The player needs to resolve 1800K/1800_complete.m3u8.

Following [RFC3986]: The m3u8 does not have a base URI embedded within it, so, Section 5.1.1 does not apply. The relative reference (and therefore the representation it references) is enclosed in an encapsulating entity, so, Section 5.1.2 applies. Section 5.1.2 says 'Thus, the default base URI of a representation is the base URI of the entity in which the representation is encapsulated.'

So what is the base URI of the encapsulating entity (/vod/movie/master.m3u8)?

1. If we follow [RFC3986] again: The master.m3u8 does not have a base URI embedded within it, so, Section 5.1.1 does not apply.

2. The master.m3u8 is not encapsulated in any other entity, so, 5.1.2 does not apply. A URI was used to retrieve the m3u8, so, Section 5.1.3 applies.

3. Section 5.1.3 says 'if a URI was used to retrieve the representation, that URI shall be considered the base URI. Note that if the retrieval was the result of a redirected request, the last URI used (i.e., the URI that resulted in the actual

retrieval of the representation) is the base URI.'

4. As the retrieval was the result of a redirected request, the second sentence applies and the last URI is https://edge1.dcdn.example.net/vod/movie/master.m3u8 and so that is the base URI for the manifest (the encapsulating entity).

5. Therefore the base URI for the relative URI 1800K/1800_complete.m3u8 is https://edge1.dcdn.example.net/vod/movie/master.m3u8 and the player should request https://edge1.dcdn.example.net/vod/movie/1800K/1800_complete.m3u8

6.2. Example 2: Relative Reference in an Encapsulating Entity

Let us take the same example, but instead of the player being passed an absolute URI, let us assume the URI is relative and embedded within another entity, i.e., a top-level HTML page https://video.cdn.example.com/index.html. In that case /vod/movie/master.m3u8 (the encapsulating entity of 1800K/1800_complete.m3u8) is itself encapsulated (in /index.html) and Section 5.1.2 does apply this time, and the base URI of /vod/movie/master.m3u8 depends on the base URI of /index.html

The base URI of /index.html (which is not encapsulated in another entity) follows Section 5.1.3, and depends on how /index.html was retrieved.

1. If /index.html was retrieved without redirection, the base URI is https://video.cdn.example.com/index.html and the base URI for the relative URI 1800K/1800_complete.m3u8 is https://video.cdn.example.com/vod/movie/master.m3u8 and the player should request https://video.cdn.example.com/vod/movie/1800K/1800_complete.m3u8

2. If /index.html was retrieved via a redirect (e.g., to https://edge1.dcdn.example.net/index.html), the base URI is https://edge1.dcdn.example.net/index.html, and the base URI for the relative URI 1800K/1800_complete.m3u8 is https://edge1.dcdn.example.net/vod/movie/master.m3u8 and the

player should request https://edge1.dcdn.example.net/vod/

movie/1800K/1800_complete.m3u8

6.3. Example 3: Relative Reference with Redirect

In this example, as in Example 2 above, the manifest URI is relative and embedded within another entity within a top-level HTML page https://video.cdn.example.com/index.html. The base URI of /index.html is an absolute URI and the request for /index.html is not redirected, but, in this case, the request for /vod/movie/master.m3u8 is redirected.

What would be resulting URI for 1800K/1800_complete.m3u8 in this case ?

1. According to Section 5.1.2 of [RFC3986] 1800K/1800_complete.m3u8 will get its base URI from the base URI of the encapsulating entity which is /vod/movie/master.m3u8.

2. As /vod/movie/master.m3u8 is itself encapsulated within /index.html then according to Section 5.1.2 its base URI is the taken from the base URI of /index.html. Since this base URI is https://video.cdn.example.com/index.html, the base URI of the master.m3u8 is resolved to https://video.cdn.example.com/vod/movie/master.m3u8, though it was redirected and its actual retrieval URI was https://edge1.dcdn.example.net/vod/movie/master.m3u8

3. The 1800K/1800_complete.m3u8, taking its base URI from the the encapsulating master.m3u8, is resolved to https://video.cdn.example.com/vod/movie/1800K/1800_complete.m3u8

One can see that in this case, as Section 5.1.2 does not define what happens if the encapsulating entity was redirected, the straightforward interpretation leads to the undesired result of the 1800_complete.m3u8 being requested from the original URI rather than the redirection URI.

7. Conclusion

The examples above show that the behavior under playlist redirect depends on how the playlist URI was resolved, and whether the playlist URI was an absolute URI or a relative one. It also depends on how exactly Section 5.1.2 of [RFC3986] is interpreted in the absence of clear directions for the case that the encapsulating entity itself was a relative reference which was redirected.

As explained above, we would like to have a single clear specification defining a consistent client behavior that is 'sticky' after a playlist redirect, regardless of how the playlist URI was resolved. We, therefore, hope that the members of HTTPbis working group could share their views regarding the preferred approach in order to resolve this issue.

8. IANA Considerations

There are no IANA considerations.

9. Security Considerations

TBC.

10. Acknowledgements

The authors would like to thank the members of the Open Caching working group of the Streaming Video Alliance (SVA) for their feedback and especially Ben Niven-Jenkins and Kevin J. Ma for their help in understanding the current state of the standards for the provided examples.

11. Contributors

TBC.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

[MPEG-DASH] ISO/IEC 23009-1:2014, "Dynamic Adaptive Streaming over HTTP-- Part 1: Media Presentation Description and Segment Formats", 2014.

[RFC6983] van Brandenburg, R., van Deventer, O., Le Faucheur, F., and K. Leung, "Models for HTTP-Adaptive-Streaming-Aware Content Distribution Network Interconnection (CDNI)", RFC 6983, DOI 10.17487/RFC6983, July 2013, <https://www.rfc-editor.org/info/rfc6983>.

[RFC8216] Pantos, R., Ed. and W. May, "HTTP Live Streaming", RFC 8216, DOI 10.17487/RFC8216, August 2017, <https://www.rfc-editor.org/info/rfc8216>.

Authors' Addresses

Ori Finkelman
Qwilt
6, Ha'harash
Hod HaSharon 4524079
Israel

Email: orif@qwilt.com

Ali Begen
Networked Media
Konya
Turkey

Email: ali.begen@networked.media