Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 9, 2016

J. Reschke
greenbytes
S. Loreto
Ericsson
October 7, 2015

'Out-Of-Band' Content Coding for HTTP
draft-reschke-http-oob-encoding-01

Abstract

This document describes an Hypertext Transfer Protocol (HTTP) content coding that can be used to describe the location of a secondary resource that contains the payload.

Editorial Note (To be removed by RFC Editor before publication)

Distribution of this document is unlimited. Although this is not a work item of the HTTPbis Working Group, comments should be sent to the Hypertext Transfer Protocol (HTTP) mailing list at ietf-http-wg@w3.org [1], which may be joined by sending a message with subject "subscribe" to ietf-http-wg-request@w3.org [2].

Discussions of the HTTPbis Working Group are archived at <http://lists.w3.org/Archives/Public/ietf-http-wg/>.

XML versions, latest edits, and issue tracking for this document are available from <https://github.com/reschke/oobencoding> and <http://greenbytes.de/tech/webdav/#draft-reschke-http-oob-encoding>.

The changes in this draft are summarized in Appendix C.1.

Status of This Memo

This Internet-Draft will expire on April 9, 2016.

Table of Contents

1. Introduction

This document describes an Hypertext Transfer Protocol (HTTP) content coding (Section 3.1.2.1 of [RFC7231]) that can be used to describe the location of a secondary resource that contains the payload.

The primary use case for this content coding is to enable origin servers to delegate the delivery of content to a secondary server that might be "closer" to the client (with respect to network topology) and/or able to cache content, leveraging content encryption, as described in [ENCRYPTENC].

2. Notational Conventions

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 [RFC2119].

This document reuses terminology used in the base HTTP specifications, namely Section 2 of [RFC7230] and Section 3 of [RFC7231].

3. 'Out-Of-Band' Content Coding

3.1. Overview

The 'Out-Of-Band' content coding is used to direct the recipient to retrieve the actual message representation (Section 3 of [RFC7231]) from a secondary resource, such as a public cache:

1. Client performs GET request

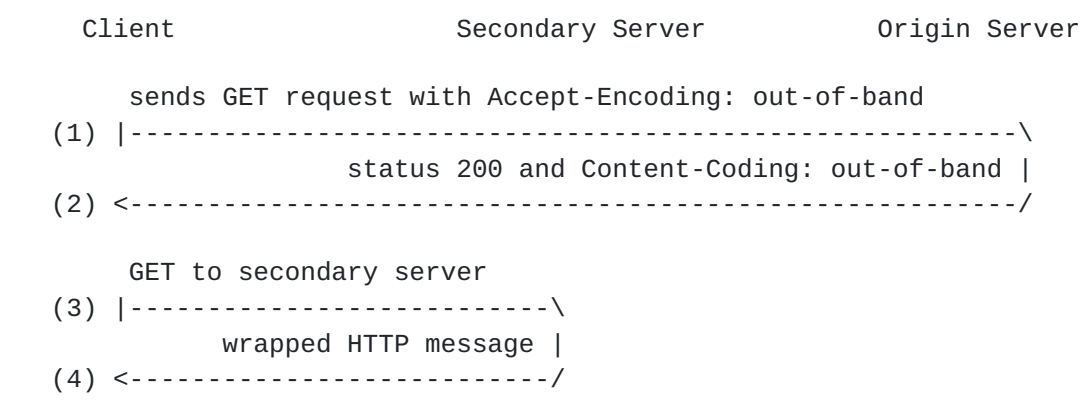
2. Received response specifies the 'out-of-band' content coding; the payload of the response contains additional meta data, plus the location of the secondary resource

3. Client performs GET request on secondary resource (usually again via HTTP(s))

4. Secondary server provides wrapped HTTP message

5. Client unwraps that representation (obtaining a full HTTP message)

6. Client combines above representation with additional representation metadata obtained from the primary resource



3.2. Definitions

The name of the content coding is "out-of-band".

The payload format uses JavaScript Object Notation (JSON, [RFC7159]), describing an array of objects describing secondary resources, each containing some of the members below:

'URI' A REQUIRED string containing the URI reference (Section 4.1 of [RFC3986]) of the secondary resource.

'metadata' An OPTIONAL object containing additional members, representing header field values to be recombined with the metadata from the secondary resource and which can not appear as header fields in the response message itself (header fields that occur multiple times need to be combined into a single field value as per Section 3.2.2 of [RFC7230]; header field names are lower-cased).

The payload format uses a JSON array so that the origin server can specify multiple secondary resources. When a client receives a response containing multiple entries, it is free to choose which of these to use.

The representation of the secondary resource needs to use a media type capable of representing a full HTTP message. For now the only supported type is "application/http" (Section 8.3.2 of [RFC7230]).

The client then obtains the original message by:

1. Unwrapping the encapsulated HTTP message by removing any transfer and content codings.

   The latter might require additional metadata that could be present in the "metadata" object, such as the "Encryption-Key" header field described in Section 4 of [ENCRYPTENC].

2. Replacing/setting any response header fields from the primary response except for framing-related information such as Content-Length, Transfer-Encoding and Content-Encoding.

3. Replacing/setting any header fields with those present as members in the "metadata" object. [[anchor3: Do we have a use case for this?]]

If the client is unable to retrieve the secondary resource's representation (host can't be reached, non 2xx response status code, payload failing integrity check, etc.), it can choose an alternate secondary resource (if specified), or simply retry the request to the origin server without including "out-of-band" in the Accept-Encoding request header field. In the latter case, it can be useful to inform the origin server about what problems were encountered when trying to access the secondary resource; see Section 3.3 for details.

Note that although this mechanism causes the inclusion of external content, it will not affect the application-level security properties of the reconstructed message, such as its web origin ([RFC6454]).

The cacheability of the response for the secondary resource does not affect the cacheability of the reconstructed response message, which is the same as for the origin server's response.

Note that because the server's response depends on the request's Accept-Encoding header field, the response usually will need to be declared to vary on that. See Section 7.1.4 of [RFC7231] and Section 2.3 of [RFC7232] for details.

3.3. Problem Reporting

When the client fails to obtain the secondary resource, it can be useful to inform the origin server about the condition. This can be accomplished by adding a "Link" header field ([RFC5988]) to a subsequent request to the origin server, detailing the URI of the secondary resource and the failure reason.

The following link extension relations are defined:

3.3.1. Server Not Reachable

Used in case the server was not reachable.

Link relation:

http://purl.org/NET/linkrel/not-reachable

3.3.2. Resource Not Found

Used in case the server responded, but the object could not be obtained.

Link relation:

http://purl.org/NET/linkrel/resource-not-found

3.3.3. Payload Unusable

Used in case the the payload could be obtained, but wasn't usable (for instance, because integrity checks failed).

Link relation:

http://purl.org/NET/linkrel/payload-unusable

3.4. Examples

3.4.1. Basic Example

Client request of primary resource:

```
GET /test HTTP/1.1
Host: www.example.com
Accept-Encoding: gzip, out-of-band
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 14 May 2015 18:52:00 GMT
Content-Type: text/plain
Cache-Control: max-age=10, public
Content-Encoding: out-of-band
Content-Length: 76
Vary: Accept-Encoding

[{
  "URI": "http://example.net/bae27c36-fa6a-11e4-ae5d-00059a3c7a00"
}]
```

(note that the Content-Type header field describes the media type of the secondary's resource representation)

Client request for secondary resource:

```
GET /bae27c36-fa6a-11e4-ae5d-00059a3c7a00 HTTP/1.1
Host: example.net
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 14 May 2015 18:52:10 GMT
Content-Type: application/http
Cache-Control: private
Content-Length: 115

HTTP/1.1 200 OK
Date: Thu, 14 May 2015 17:00:00 GMT
Content-Length: 15
Content-Language: en

Hello, world.
```

Final message after recombining header fields:

```
HTTP/1.1 200 OK
Date: Thu, 14 May 2015 18:52:00 GMT
Content-Length: 15
Cache-Control: max-age=10, public
Content-Type: text/plain
Content-Language: en

Hello, world.
```

In this example, Cache-Control, Content-Length, and Date have been set/overwritten with data from the primary resource's representation.

3.4.2. Example involving an encrypted resource

Given the example HTTP message from Section 5.4 of [ENCRYPTENC], a primary resource could use the "out-of-band" encoding to specify just the location of the secondary resource plus the contents of the "Encryption-Key" header field needed to decrypt the payload:

Response:

```
HTTP/1.1 200 OK
Date: Thu, 14 May 2015 18:52:00 GMT
Content-Encoding: out-of-band
Content-Type: text/plain
Content-Length: 192
Vary: Accept-Encoding

[{
  "URI": "http://example.net/bae27c36-fa6a-11e4-ae5d-00059a3c7a00"
  "metadata": {
    "encryption-key": "keyid=\"a1\";
                      key=\"9Z57YCb3dK95dSsdFJbkag\""
  }
}]
```

(note that the Content-Type header field describes the media type of the secondary's resource representation)

Response for secondary resource:

```
HTTP/1.1 200 OK
Date: Thu, 14 May 2015 18:52:10 GMT
Content-Type: application/http
Content-Length: ...
Cache-Control: private

HTTP/1.1 200 OK
Content-Length: 31
Content-Encoding: aesgcm128
Encryption: keyid="a1"; salt="ibZx1RNz537h1XNkRcPpjA"

zK3kpG__Z8whjIkG6RYgPz11oUkTKcxPy9WP-VPMfuc
```

(payload body shown in base64 here)

Final message after recombining header fields:

```
HTTP/1.1 200 OK
Date: Thu, 14 May 2015 18:52:00 GMT
Content-Length: 15
Content-Type: text/plain

I am the walrus
```

3.4.3. Example For Problem Reporting

Client requests primary resource as in Section 3.4.1, but the attempt to access the secondary resource fails.

Response:

```
HTTP/1.1 404 Not Found
Date: Thu, 08 September 2015 16:49:00 GMT
Content-Type: text/plain
Content-Length: 20

Resource Not Found
```

Client retries with the origin server and includes Link header field reporting the problem:

```
GET /test HTTP/1.1
Host: www.example.com
Accept-Encoding: gzip, out-of-band
Link: <http://example.net/bae27c36-fa6a-11e4-ae5d-00059a3c7a00>;
      rel="http://purl.org/NET/linkrel/resource-not-found"
```

4. Feature Discovery

New content codings can be deployed easily, as the client can use the "Accept-Encoding" header field (Section 5.3.4 of [RFC7231]) to signal which content codings are supported.

5. Security Considerations

5.1. Content Modifications

This specification does not define means to verify that the payload obtained from the secondary resource really is what the origin server expects it to be. Content signatures can address this concern (see [CONTENTSIG]).

5.2. Use in Requests

In general, content codings can be used in both requests and responses. This particular content coding has been designed for responses. When supported in requests, it creates a new attack vector where the receiving server can be tricked into including content that the client might not have access to otherwise (such as HTTP resources behind a firewall).

6. IANA Considerations

The IANA "HTTP Content Coding Registry", located at <http://www.iana.org/assignments/http-parameters>, needs to be updated with the registration below:

Name: out-of-band

Description: Payload needs to be retrieved from a secondary resource

Reference: Section 3 of this document

7. References

7.1. 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, <http://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,

<http://www.rfc-editor.org/info/rfc3986>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://www.rfc-editor.org/info/rfc5988>.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

7.2. Informative References

[CONTENTSIG] Thomson, M., "Content-Signature Header Field for HTTP", draft-thomson-http-content-signature-00 (work in progress), July 2015.

[ENCRYPTENC] Thomson, M., "Encrypted Content-Encoding for HTTP", draft-thomson-http-encryption-01 (work in progress), July 2015.

[RFC2017] Freed, N. and K. Moore, "Definition of the URL MIME External-Body Access-Type", RFC 2017, DOI 10.17487/RFC2017, October 1996, <http://www.rfc-editor.org/info/rfc2017>.

[RFC4483] Burger, E., "A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages", RFC 4483, DOI 10.17487/RFC4483, May 2006, <http://www.rfc-editor.org/info/rfc4483>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc6454>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <http://www.rfc-editor.org/info/rfc7232>.

URIs

[1] <mailto:ietf-http-wg@w3.org>

[2] <mailto:ietf-http-wg-request@w3.org?subject=subscribe>

Appendix A. Alternatives, or: why not a new Status Code?

A plausible alternative approach would be to implement this functionality one level up, using a new redirect status code (Section 6.4 of [RFC7231]). However, this would have several drawbacks:

- Servers will need to know whether a client understands the new status code; thus some additional signal to opt into this protocol would always be needed.

- In redirect messages, representation metadata (Section 3.1 of [RFC7231]), namely "Content-Type", applies to the response message, not the redirected-to resource.

- The origin-preserving nature of using a content coding woudld be lost.

Another alternative would be to implement the indirection on the level of the media type using something similar to the type "message/external-body", defined in [RFC2017] and refined for use in the Session Initiation Protocol (SIP) in [RFC4483]. This approach though would share most of the drawbacks of the status code approach mentioned above.

Appendix B. Open Issues

B.1. Range Requests

We probably need to handle Range Requests. How would this work? Passing down the Range request header field to the secondary resource?

What about codes other than 200 and 206?

B.2. Accessing the Secondary Resource Too Early

One use-case for this protocol is to enable a system of "blind caches", which would serve the secondary resources. These caches might only be populated on demand, thus it could happen that whatever mechanism is used to populate the cache hasn't finished when the client hits it (maybe due to race conditions, or because the cache is behind a middlebox which doesn't allow the origin server to push

content to it).

In this particular case, it can be useful if the client was able to "piggyback" the URI of the primary resource, giving the secondary server a means by which it could obtain the payload itself. This information could be provided in yet another Link header field:

```
GET bae27c36-fa6a-11e4-ae5d-00059a3c7a00 HTTP/1.1
Host: example.net
Link: <http://example.com/test>;
      rel="http://purl.org/NET/linkrel/primary-resource"
```

(continuing the example from Section 3.4.1)

What's unclear is whether it's ok for the client to reveal the URI if the primary resource, and under which conditions it's ok for the secondary server to access it. All it needs is the potentially encrypted payload, so maybe yet another URI on the origin server is needed.

Appendix C. Change Log (to be removed by RFC Editor before publication)

C.1. Changes since draft-reschke-http-oob-encoding-00

Mention media type approach.

Explain that clients can always fall back not to use oob when the secondary resource isn't available.

Add Vary response header field to examples and mention that it'll usually be needed (<https://github.com/reschke/oobencoding/issues/6>).

Experimentally add problem reporting using piggy-backed Link header fields (<https://github.com/reschke/oobencoding/issues/7>).

Appendix D. Acknowledgements

Thanks to Christer Holmberg, Daniel Lindstrom, Goran Eriksson, John Mattsson, Kevin Smith, Mark Nottingham, Martin Thomson, and Roland Zink for feedback on this document.

Authors' Addresses

Julian F. Reschke
greenbytes GmbH
Hafenweg 16
Muenster, NW 48155
Germany

EMail: julian.reschke@greenbytes.de
URI: http://greenbytes.de/tech/webdav/

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: salvatore.loreto@ericsson.com