

Network Working Group  
Internet-Draft  
Updates: [7234](#) (if approved)  
Intended status: Standards Track  
Expires: May 3, 2018

M. Nottingham  
Fastly  
October 30, 2017

HTTP Representation Variants  
draft-nottingham-variants-01

## Abstract

This specification introduces the HTTP "Variants" response header field to communicate what representations are available for a given resource, and the companion "Variant-Key" response header field to indicate which representation was selected. It is an augmentation of the "Vary" mechanism in HTTP caching.

## Note to Readers

\_RFC EDITOR: please remove this section before publication\_

The issues list for this draft can be found at <https://github.com/mnot/I-D/labels/variants> [1].

The most recent (often, unpublished) draft is at <https://mnot.github.io/I-D/variants/> [2].

Recent changes are listed at <https://github.com/mnot/I-D/commits/gh-pages/variants> [3].

See also the draft's current status in the IETF datatracker, at <https://datatracker.ietf.org/doc/draft-nottingham-variants/> [4].

## 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."

Internet-Draft

HTTP Representation Variants

October 2017

This Internet-Draft will expire on May 3, 2018.

## Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](https://trustee.ietf.org/license-info) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">1.1.</a>	Notational Conventions . . . . .	<a href="#">4</a>
<a href="#">2.</a>	The "Variants" HTTP Header Field . . . . .	<a href="#">4</a>
<a href="#">2.1.</a>	Relationship to Vary . . . . .	<a href="#">6</a>
<a href="#">3.</a>	The "Variant-Key" HTTP Header Field . . . . .	<a href="#">6</a>
<a href="#">4.</a>	Defining Content Negotiation Using Variants . . . . .	<a href="#">7</a>
<a href="#">5.</a>	Cache Behaviour . . . . .	<a href="#">7</a>
<a href="#">5.1.</a>	Find Available Keys . . . . .	<a href="#">9</a>
<a href="#">5.2.</a>	Example of Cache Behaviour . . . . .	<a href="#">9</a>
<a href="#">6.</a>	Example Headers . . . . .	<a href="#">10</a>
<a href="#">6.1.</a>	Single Variant . . . . .	<a href="#">10</a>
<a href="#">6.2.</a>	Multiple Variants . . . . .	<a href="#">11</a>
<a href="#">6.3.</a>	Partial Coverage . . . . .	<a href="#">11</a>
<a href="#">7.</a>	IANA Considerations . . . . .	<a href="#">12</a>
<a href="#">8.</a>	Security Considerations . . . . .	<a href="#">13</a>
<a href="#">9.</a>	Acknowledgments . . . . .	<a href="#">13</a>
<a href="#">10.</a>	References . . . . .	<a href="#">13</a>
<a href="#">10.1.</a>	Normative References . . . . .	<a href="#">13</a>
<a href="#">10.2.</a>	Informative References . . . . .	<a href="#">14</a>
<a href="#">10.3.</a>	URIs . . . . .	<a href="#">14</a>
<a href="#">Appendix A.</a>	Variants for Existing Content Negotiation Mechanisms	<a href="#">14</a>
<a href="#">A.1.</a>	Accept-Encoding . . . . .	<a href="#">14</a>
<a href="#">A.2.</a>	Accept-Language . . . . .	<a href="#">15</a>
	Author's Address . . . . .	<a href="#">15</a>

## 1. Introduction

HTTP proactive content negotiation ([\[RFC7231\]](#), [Section 3.4.1](#)) is starting to be used more widely again. The most widely seen use - determining a response's content-coding - is being joined by renewed interest in negotiation for language and other, newer attributes (for example, see [\[I-D.ietf-httpbis-client-hints\]](#)).

Successfully reusing negotiated responses that have been stored in a HTTP cache requires establishment of a secondary cache key ([\[RFC7234\]](#), [Section 4.1](#)) using the Vary header ([\[RFC7231\]](#), [Section 7.1.4](#)), which identifies the request headers that form the secondary cache key for a given response.

HTTP's caching model allows a certain amount of latitude in normalising request header fields identified by Vary to match those stored in the cache, so as to increase the chances of a cache hit while still respecting the semantics of that header. However, this is often inadequate; even with understanding of the headers' semantics to facilitate such normalisation, a cache does not know enough about the possible alternative representations available on the origin server to make an appropriate decision.

For example, if a cache has stored the following request/response pair:

```
GET /foo HTTP/1.1
Host: www.example.com
Accept-Language: en;q=1.0, fr;q=0.5
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Language: fr
Vary: Accept-Language
Transfer-Encoding: chunked
```

[French content]

Provided that the cache has full knowledge of the semantics of Accept-Language and Content-Language, it will know that a French representation is available and might be able to infer that an English representation is not available. But, it does not know (for example) whether a Japanese representation is available without making another request, thereby incurring possibly unnecessary latency.

This specification introduces the HTTP Variants response header field ([Section 2](#)) to enumerate the available variant representations on the

Nottingham

Expires May 3, 2018

[Page 3]

---

Internet-Draft

HTTP Representation Variants

October 2017

origin server, to provide clients and caches with enough information to properly satisfy requests - either by selecting a response from cache or by forwarding the request towards the origin - by following an algorithm defined in [Section 5](#).

Its companion the Variant-Key response header field ([Section 3](#)) indicates which representation was selected, so that it can be reliably reused in the future.

This mechanism requires that proactive content negotiation mechanisms define how they use it; see [Section 4](#). It is best suited for negotiation over request headers that are well-understood. It also works best when content negotiation takes place over a constrained set of representations; since each variant needs to be listed in the header field, it is ill-suited for open-ended sets of representations.

It can be seen as a simpler version of the Alternates header field introduced by [[RFC2295](#)]; unlike that mechanism, Variants does not require specification of each combination of attributes, and does not assume that each combination has a unique URL.

### [1.1](#). Notational Conventions

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.

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [\[RFC5234\]](#) with a list extension, defined in [Section 7 of \[RFC7230\]](#), that allows for compact definition of comma-separated lists using a '#' operator (similar to how the '\*' operator indicates repetition).

Additionally, it uses the "field-name", "OWS" and "token" rules from [\[RFC7230\]](#).

## 2. The "Variants" HTTP Header Field

The Variants HTTP response header field indicates what representations are available for a given resource at the time that the response is produced, by enumerating the request header fields that it varies on, along with the values that are available for each.

```
Variants      = 1#variant-item
variant-item   = field-name *( OWS ";" OWS available-value )
available-value = token
```

Each "variant-item" indicates a request header field that carries a value that clients might proactively negotiate for; each parameter on it indicates a value for which there is an available representation on the origin server.

So, given this example header field:

Variants: Accept-Encoding;gzip

a recipient can infer that the only content-coding available for that resource is "gzip" (along with the "identity" non-encoding; see [Appendix A.1](#)).

Given:

Variants: accept-encoding

a recipient can infer that no content-codings (beyond identity) are

supported. Note that as always, field-name is case-insensitive.

A more complex example:

Variants: Accept-Encoding;gzip;br, Accept-Language;en ;fr

Here, recipients can infer that two content-codings in addition to "identity" are available, as well as two content languages. Note that, as with all HTTP header fields that use the "#" list rule (see [\[RFC7230\], Section 7](#)), they might occur in the same header field or separately, like this:

Variants: Accept-Encoding;gzip;brotli

Variants: Accept-Language;en ;fr

The ordering of available-values after the field-name is significant, as it might be used by the header's algorithm for selecting a response (see [Appendix A.1](#) for an example of this).

The ordering of the request header fields themselves indicates descending application of preferences; for example, in the headers above, a cache will serve gzip'd content regardless of language if it is available.

Origin servers SHOULD consistently send Variant header fields on all cacheable (as per [\[RFC7234\], Section 3](#)) responses for a resource,

since its absence will trigger caches to fall back to Vary processing.

Likewise, servers MUST send the Variant-Key response header field when sending Variants.

### [2.1.](#) Relationship to Vary

Caches that fully implement this specification SHOULD ignore request header fields in the "Vary" header for the purposes of secondary cache key calculation ([\[RFC7234\], Section 4.1](#)) when their semantics are implemented as per this specification and their corresponding response header field is listed in "Variants".

If any member of the Vary header does not have a corresponding

variant that is understood by the implementation, it is still subject to the requirements there.

### [3.](#) The "Variant-Key" HTTP Header Field

The Variant-Key HTTP response header field is used to indicate the value(s) from the Variants header field that identify the representation it occurs within.

Variant-Key        = 1#available-value

Each value indicates the selected available-value, in the same order as the variants listed in the Variants header field.

Therefore, Variant-Key MUST be the same length (in comma-separated members) as Variants, and each member MUST correspond in position to its companion in Variants.

For example:

Variants: Content-Encoding;gzip;br, Content-Language;en ;fr  
Variant-Key: gzip, fr

This header pair indicates that the representation is used for responses that have a "gzip" content-coding and "fr" content-language.

Note that the contents of Variant-Key are only used to indicate what request attributes are identified with the response containing it; this is different from headers like Content-Encoding, which indicate attributes of the response. In the example above, it might be that a gzip'd version of the French content is not available, in which case

it will not include "Content-Encoding: gzip", but still have "gzip" in Variant-Key.

### [4.](#) Defining Content Negotiation Using Variants

To be usable with Variants, proactive content negotiation mechanisms need to be specified to take advantage of it. Specifically, they:

- o MUST define a request header field that advertises the clients preferences or capabilities, whose field-name SHOULD begin with "Accept-".
- o MUST define the syntax of available-values that will occur in Variants and Variant-Key.
- o MUST define an algorithm for selecting a result. It MUST return a list of available-values that are suitable for the request, in order of preference, given the value of the request header nominated above and an available-values list from the Variants header. If the result is an empty list, it implies that the cache cannot satisfy the request.

[Appendix A](#) fulfils these requirements for some existing proactive content negotiation mechanisms in HTTP.

## 5. Cache Behaviour

Caches that implement the Variants header field and the relevant semantics of the field-name it contains can use that knowledge to either select an appropriate stored representation, or forward the request if no appropriate representation is stored.

They do so by running this algorithm (or its functional equivalent) upon receiving a request, incoming-request:

1. Let selected-responses be a list of the stored responses suitable for reuse as defined in [\[RFC7234\] Section 4](#), excepting the requirement to calculate a secondary cache key.
2. Order selected-responses by the "Date" header field, most recent to least recent.
3. If the freshest (as per [\[RFC7234\], Section 4.2](#)) has one or more "Variants" header field(s):
  1. Select one member of selected\_responses and let its "Variants" header field-value(s) be variants-header. This



one as long as it is still fresh.

2. Let sorted-variants be an empty list.
3. For each variant in variants-header:
  1. If variant's field-name corresponds to the response header field identified by a content negotiation mechanism that the implementation supports:
    1. Let request-value be the field-value of the request header field(s) identified by the content negotiation mechanism.
    2. Let available-values be a list containing all available-value for the variant.
    3. Let sorted-values be the result of running the algorithm defined by the content negotiation mechanism with request-value and available-values.
    4. Append sorted-values to sorted-variants.

At this point, sorted-variants will be a list of lists, each member of the top-level list corresponding to a variant-item in the Variants header field-value, containing zero or more items indicating available-values that are acceptable to the client, in order of preference, greatest to least.

4. If any member of sorted-variants is an empty list, stop processing and forward the request towards the origin, since an acceptable response is not stored in the cache.
5. Let sorted-keys be the result of running Find Available Keys ([Section 5.1](#)) on sorted-variants and two empty lists.

This will result in a list of lists, where each member of the top-level list indicates, in preference order, a key for an acceptable response to the request.

A Cache MAY satisfy the request with any response whose Variant-Key header corresponds to a member of sorted-keys; when doing so, it SHOULD use the most preferred available response.

See also [Section 2.1](#) regarding handling of Vary.

### [5.1.](#) Find Available Keys

Given sorted-variants, a list of lists, and key-stub, a list representing a partial key, and possible-keys, a list:

1. Let sorted-values be the first member of sorted-variants.
2. For each sorted-value in sorted-values:
  1. Let this-key be a copy of key-stub.
  2. Append sorted-value to this-key.
  3. Let remaining-variants be a copy of all of the members of sorted-variants except the first.
  4. If remaining-variants is empty, append this-key to possible-keys.
  5. Else, run Find Available Keys on remaining-variants, this-key and possible-keys.
  6. Return possible-keys.

### [5.2.](#) Example of Cache Behaviour

For example, if the selected variants-header was:

Variants: Accept-Language;en;fr,de, Accept-Encoding;gzip,br

and the request contained the headers:

Accept-Language: fr;q=1.0, en;q=0.1  
Accept-Encoding: gzip

Then the sorted-variants would be:

```
[  
  ["fr", "en"]           // prefers French, will accept English  
  ["gzip", "identity"]   // prefers gzip encoding, will accept identity  
]
```

Which means that the sorted-keys would be:

Internet-Draft

HTTP Representation Variants

October 2017

```
[  
  ['fr', 'gzip'],  
  ['fr', 'identity'],  
  ['en', 'gzip'],  
  ['en', 'identity']  
]
```

Representing a first preference of a French, gzip'd response. Thus, if a cache has a response with:

Variant-Key: fr, gzip

it could be used to satisfy the first preference. If not, responses corresponding to the other keys could be returned, or the request could be forwarded towards the origin.

## [6.](#) Example Headers

### [6.1.](#) Single Variant

Given a request/response pair:

```
GET /foo HTTP/1.1  
Host: www.example.com  
Accept-Language: en;q=1.0, fr;q=0.5
```

```
HTTP/1.1 200 OK  
Content-Type: image/gif  
Content-Language: en  
Cache-Control: max-age=3600  
Variants: Content-Language;en;de  
Variant-Key: en  
Vary: Accept-Language  
Transfer-Encoding: chunked
```

Upon receipt of this response, the cache knows that two representations of this resource are available, one with a "Content-Language" of "en", and another whose "Content-Language" is "de".

Subsequent requests (while this response is fresh) will cause the cache to either reuse this response or forward the request, depending on what the selection algorithm determines.

So, if a request with "en" in "Accept-Language" is received and its q-value indicates that it is acceptable, the stored response is used. A request that indicates that "de" is acceptable will be forwarded to the origin, thereby populating the cache. A cache receiving a

request that indicates both languages are acceptable will use the q-value to make a determination of what response to return.

A cache receiving a request that does not list either language as acceptable (or does not contain an Accept-Language at all) will return the "en" representation (possibly fetching it from the origin), since it is listed first in the "Variants" list.

Note that "Accept-Language" is listed in Vary, to assure backwards-compatibility with caches that do not support "Variants".

## [6.2.](#) Multiple Variants

A more complicated request/response pair:

```
GET /bar HTTP/1.1
Host: www.example.net
Accept-Language: en;q=1.0, fr;q=0.5
Accept-Encoding: gzip, br

HTTP/1.1 200 OK
Content-Type: image/gif
Content-Language: en
Content-Encoding: br
Variants: Content-Language;en;jp;de
Variants: Content-Encoding;br;gzip
Variant-Key: en, br
Vary: Accept-Language, Accept-Encoding
Transfer-Encoding: chunked
```

Here, the cache knows that there are two axes that the response varies upon; "Content-Language" and "Content-Encoding". Thus, there

are a total of six possible representations for the resource, and the cache needs to consider the selection algorithms for both axes.

Upon a subsequent request, if both selection algorithms return a stored representation, it can be served from cache; otherwise, the request will need to be forwarded to origin.

### [6.3.](#) Partial Coverage

Now, consider the previous example, but where only one of the Vary'd axes is listed in "Variants":

Nottingham

Expires May 3, 2018

[Page 11]

---

Internet-Draft

HTTP Representation Variants

October 2017

```
GET /bar HTTP/1.1
Host: www.example.net
Accept-Language: en;q=1.0, fr;q=0.5
Accept-Encoding: gzip, br
```

```
HTTP/1.1 200 OK
Content-Type: image/gif
Content-Language: en
Content-Encoding: br
Variants: Content-Encoding;br;gzip
Variant-Key: br
Vary: Accept-Language, Accept-Encoding
Transfer-Encoding: chunked
```

Here, the cache will need to calculate a secondary cache key as per [\[RFC7234\]](#), [Section 4.1](#) - but considering only "Accept-Language" to be in its field-value - and then continue processing "Variants" for the set of stored responses that the algorithm described there selects.

## [7.](#) IANA Considerations

This specification registers two values in the Permanent Message Header Field Names registry established by [\[RFC3864\]](#):

- o Header field name: Variants

- o Applicable protocol: http
- o Status: standard
- o Author/Change Controller: IETF
- o Specification document(s): [this document]
- o Related information:
- o Header field name: Variant-Key
- o Applicable protocol: http
- o Status: standard
- o Author/Change Controller: IETF
- o Specification document(s): [this document]
- o Related information:

## [8.](#) Security Considerations

If the number or advertised characteristics of the representations available for a resource are considered sensitive, the "Variants" header by its nature will leak them.

Note that the "Variants" header is not a commitment to make representations of a certain nature available; the runtime behaviour of the server always overrides hints like "Variants".

## [9.](#) Acknowledgments

This protocol is conceptually similar to, but simpler than, Transparent Content Negotiation [[RFC2295](#)]. Thanks to its authors for their inspiration.

It is also a generalisation of a Fastly VCL feature designed by Rogier 'DocWilco' Mulhuijzen.

Thanks to Hooman Beheshti for his review and input.

## 10. References

### 10.1. 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>>.
- [RFC4647] Phillips, A. and M. Davis, "Matching of Language Tags", [BCP 47](#), [RFC 4647](#), DOI 10.17487/RFC4647, September 2006, <<https://www.rfc-editor.org/info/rfc4647>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, [RFC 5234](#), DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [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, <<https://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, <<https://www.rfc-editor.org/info/rfc7231>>.

Nottingham

Expires May 3, 2018

[Page 13]

---

Internet-Draft

HTTP Representation Variants

October 2017

- [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", [RFC 7234](#), DOI 10.17487/RFC7234, June 2014, <<https://www.rfc-editor.org/info/rfc7234>>.
- [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>>.

### 10.2. Informative References

[I-D.ietf-httpbis-client-hints]

Grigorik, I., "HTTP Client Hints", [draft-ietf-httpbis-client-hints-04](#) (work in progress), April 2017.

[RFC2295] Holtman, K. and A. Mutz, "Transparent Content Negotiation in HTTP", [RFC 2295](#), DOI 10.17487/RFC2295, March 1998, <<https://www.rfc-editor.org/info/rfc2295>>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", [BCP 90](#), [RFC 3864](#), DOI 10.17487/RFC3864, September 2004, <<https://www.rfc-editor.org/info/rfc3864>>.

### [10.3.](#) URIs

[1] <https://github.com/mnot/I-D/labels/variants>

[2] <https://mnot.github.io/I-D/variants/>

[3] <https://github.com/mnot/I-D/commits/gh-pages/variants>

[4] <https://datatracker.ietf.org/doc/draft-nottingham-variants/>

## [Appendix A.](#) Variants for Existing Content Negotiation Mechanisms

This appendix defines the required information to use existing proactive content negotiation mechanisms (as defined in [\[RFC7231\]](#), [Section 5.3](#)) with the "Variants" header field.

### [A.1.](#) Accept-Encoding

This section defines handling for "Accept-Encoding" variants, as per [\[RFC7231\]](#) [Section 5.3.4](#).

To perform content negotiation for Accept-Encoding given an request-value and available-values:

1. Let preferred-codings be a list of the codings in the request-value, ordered by their weight, highest to lowest, as per [\[RFC7231\]](#) [Section 5.3.1](#) (omitting any coding with a weight of 0). If "Accept-Encoding" is not present or empty, preferred-codings



will be empty.

2. If "identity" is not a member of preferred-codings, append "identity".
3. Append "identity" to available-values.
4. Remove any member of available-values not present in preferred-codings, comparing in a case-insensitive fashion.
5. Return available-values.

#### [A.2.](#) Accept-Language

This section defines handling for "Accept-Language" variants, as per [\[RFC7231\] Section 5.3.5](#).

To perform content negotiation for Accept-Language given an request-value and available-values:

1. Let preferred-langs be a list of the language-ranges in the request-value, ordered by their weight, highest to lowest, as per [\[RFC7231\] Section 5.3.1](#) (omitting any language-range with a weight of 0).
2. If preferred-langs is empty, append "\*".
3. Filter available-values using preferred-langs with either the Basic Filtering scheme defined in [\[RFC4647\] Section 3.3.1](#), or the Lookup scheme defined in [Section 3.4](#) of that document. Use the first member of available-values as the default.
4. Return available-values.

#### Author's Address

Mark Nottingham  
Fastly

Email: [mnot@mnot.net](mailto:mnot@mnot.net)  
URI: <https://www.mnot.net/>