Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 18, 2013

R. Fielding
Adobe Systems Incorporated
M. Nottingham

October 15, 2012

The Key HTTP Response Header Field
draft-fielding-http-key-01

Abstract

The 'Key' header field for HTTP responses allows an origin server to describe the cache key for a negotiated response: a short algorithm that can be used upon later requests to determine if the same response is reusable.

Key has the advantage of avoiding an additional round trip for validation whenever a new request differs slightly, but not significantly, from prior requests.

Key also informs user agents of the request characteristics that might result in different content, which can be useful if the user agent is not sending Accept* fields in order to reduce the risk of fingerprinting.

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 http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 2013.

Table of Contents

1. Introduction

In HTTP caching [I-D.ietf-httpbis-p6-cache], the Vary response header field effectively modifies the key used to store and access a response to include information from the request's headers. This allows proactive content negotiation [I-D.ietf-httpbis-p2-semantics] to work with caches.

However, Vary's operation is coarse-grained; although caches are allowed to normalise the values of headers based upon their semantics, doing so requires the cache to understand those semantics, and is still quite limited.

For example, if a response is cached with the response header field:

```
Vary: Accept-Encoding
```

and and its associated request is:

```
Accept-Encoding: gzip
```

then a subsequent request with the following header is (in a strict reading of HTTP) not a match, resulting in a cache miss:

```
Accept-Encoding: identity, gzip
```

This document defines a new response header field, "Key", that allows servers to describe the cache key in a much more fine-grained manner, leading to improved cache efficiency.

1.1. 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 uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234] with the list rule extension defined in [I-D.ietf-httpbis-p1-messaging], Appendix B. It includes by reference the OWS, field-name and quoted-string rules from that document, and the parameter rule from [I-D.ietf-httpbis-p2-semantics].

2. The "Key" Response Header Field

The "Key" response header field describes the request attributes that the server has used to select the current representation.

As such, its semantics are similar to the "Vary" response header field, but it allows more fine-grained description, using "key modifiers".

Caches can use this information as part of determining whether a stored response can be used to satisfy a given request. When a cache fully implements this mechanism, it MAY ignore the Vary response header field.

Additionally, user agents can use this information to discover if additional request header fields might influence the resulting response.

The Key field-value is a comma-delimited list of selecting header fields (similar to Vary), with zero to many modifiers to each, delimited by semicolons. Whitespace is not allowed in the field-value between each field-name and its parameter set.

```
Key = 1#field-name *( ";" parameter )
```

The following header fields therefore have identical semantics:

```
Vary: Accept-Encoding, Accept-Language
Key: Accept-Encoding, Accept-Language
```

However, Key's use of modifiers allows:

```
Key: Accept-Encoding;w="gzip", Accept-Language;b="fr"
```

to indicate that the response it occurs in is allowed to be reused for requests that contain the token "gzip" (in any case) in the Accept-Encoding header field and an Accept-Language header field value that starts with "fr" (in any case).

Note that both the field-name and modifier names themselves are case insensitive.

2.1. Header Matching

When used by a cache to determine whether a stored response can be used to satisfy a presented request, each field-name identifies a potential request header, just as with the Vary response header field.

However, each of these can have zero to many "key modifiers" that change how the response selection process (as defined in [I-D.ietf-httpbis-p6-cache], Section 4.3)) works.

In particular, a cache that implements the Key header field MUST NOT use a stored response unless all of the selecting header fields nominated by the Key header field match in both the original request (i.e., that associated with the stored response) and the presented request.

Modifiers operate on a list of zero to many field-values. This list is constructed by:

1. Starting with the field-values of all header fields that have the given field-name.
2. Splitting each field-value on commas, excluding any that occur inside of a quoted-string production.
3. Flattening the list of lists into a single list that represents the individual header field-values.
4. Case-normalising each value in both lists to lowercase.
5. Trimming any OWS from the start and end of the field-values.

For example, given the set of headers:

```
Foo: 1
Bar: z
Foo: 2, a="b,c"
```

A modifier for "Foo" would operate on the list of presented values '1', '2', 'a="b,c"'.

Note that step 2 of this algorithm optimistically assumes that double-quotes in a header field value denote the use of the quoted-string as defined by [I-D.ietf-httpbis-p1-messaging]; the cache does not need to "understand" the specific header field.

Once the appropriate header fields from both the original request and the stored request are processed in this manner, the result is two (possibly empty) lists of values for each specified header field.

The key modifiers (as specified in the Key header field) are then applied to the lists in the order they appear in Key (left to right). If any modifier does not return a match (as per its definition), the headers are said not to match. If all of the modifiers return a match, the headers are said to match.

Note that some types of modifiers are said to always match; they can be used to alter the input lists, or to alter the semantics of subsequent matches.

Unrecognised modifiers MUST result in a failure to match.

2.2. Key Modifiers

This document defines the following key modifiers:

2.2.1. "w": Word Match Modifier

The "w" modifier matches if the parameter value (after unquoting) matches (character-for-character) any whole value in both lists.

2.2.2. "s": Substring Match Modifier

The "s" modifier matches if the parameter value (after unquoting) is contained as a sequence of characters within both lists.

2.2.3. "b": Beginning Substring Match Modifier

The "b" modifier matches if both lists contain a value that begins with the same sequence of characters as the parameter value (after unquoting).

2.2.4. "p": Parameter Prefix Match Modifier

The "p" modifier matches if the parameter value (after unquoting) matches (character-for-character) the sequence of characters up to (but not including) the first semi-colon (";") in both lists, after any whitespace is removed.

For example, given the key:

```
Key: Accept;p="text/html"
```

then each of the following header fields is a match:

```
Accept: text/html
Accept: text/html; q=0.5
Accept: text/html;q=0.1
Accept: text/html; foo="bar"
```

2.2.5. "c": Case Sensitivity Flag

The "c" modifier always matches, and has the side effect of reverting the case normalisation of the header lists (see #4 in the list above), so that subsequent matches become case sensitive.

2.2.6. "n": Not Flag

The "n" modifier always matches, and has the side effect of modifying the semantics of subsequent modifiers (i.e., the match modifiers to

its right, lexically) so that they will be considered to match if they do not, and likewise they will be considered not to match if they do.

For example, given a response with:

```
Key: Foo;w="a";n;w="b"
```

then the presented header:

```
Foo: a, c
```

would match, while

```
Foo: a, b
```

would not (because it contains "b").

2.3. Examples

For example, this response header field:

```
Key: cookie;w="_sess=fhd378";c;w="ID=\"Roy\"",
     Accept-Encoding;w="gzip"
```

would allow the cache to reuse the response it occurs in if the presented request contains:

- A Cookie header containing both ID="Roy" (in that case) and _sess=fhd378 (evaluated case insensitively), and
- An Accept-Encoding header containing the token "gzip" (evaluated case insensitively).

Less convoluted examples include matching any request with a User-Agent field value containing "MSIE" in any combination of case:

```
Key: user-agent;s="MSIE"
```

And an Accept-Language field value for French:

```
Key: accept-language;b="fr"
```

3. IANA Considerations

TBD

4. Security Considerations

TBD

5. Normative References

[I-D.ietf-httpbis-p1-messaging]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", draft-ietf-httpbis-p1-messaging-21 (work in progress), October 2012.

[I-D.ietf-httpbis-p2-semantics]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", draft-ietf-httpbis-p2-semantics-21 (work in progress), October 2012.

[I-D.ietf-httpbis-p6-cache]
Fielding, R., Nottingham, M., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", draft-ietf-httpbis-p6-cache-21 (work in progress), October 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

Authors' Addresses

Roy T. Fielding
Adobe Systems Incorporated

Email: fielding@gbiv.com
URI: http://roy.gbiv.com/

Mark Nottingham

Email: mnot@mnot.net
URI: http://www.mnot.net/