Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 25 October 2022

M. Nottingham
23 April 2022

Retrofit Structured Fields for HTTP
draft-ietf-httpbis-retrofit-01

Abstract

This specification defines how a selection of existing HTTP fields can be handled as Structured Fields.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-httpbis-retrofit/.

Discussion of this document takes place on the HTTP Working Group mailing list (mailto:ietf-http-wg@w3.org), which is archived at https://lists.w3.org/Archives/Public/ietf-http-wg/. Working Group information can be found at https://httpwg.org/.

Source for this draft and an issue tracker can be found at https://github.com/httpwg/http-extensions/labels/retrofit.

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 25 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Structured Field Values for HTTP [STRUCTURED-FIELDS] introduced a data model with associated parsing and serialization algorithms for use by new HTTP field values. Header fields that are defined as Structured Fields can realise a number of benefits, including:

* Improved interoperability and security: precisely defined parsing and serialisation algorithms are typically not available for fields defined with just ABNF and/or prose.

* Reuse of common implementations: many parsers for other fields are specific to a single field or a small family of fields

* Canonical form: because a deterministic serialisation algorithm is defined for each type, Structure Fields have a canonical representation

* Enhanced API support: a regular data model makes it easier to expose field values as a native data structure in implementations

* Alternative serialisations: While [STRUCTURED-FIELDS] defines a textual serialisation of that data model, other, more efficient serialisations of the underlying data model are also possible.

However, a field needs to be defined as a Structured Field for these benefits to be realised. Many existing fields are not, making up the bulk of header and trailer fields seen in HTTP traffic on the internet.

This specification defines how a selection of existing HTTP fields can be handled as Structured Fields, so that these benefits can be realised -- thereby making them Retrofit Structured Fields.

It does so using two techniques. Section 2 lists compatible fields -- those that can be handled as if they were Structured Fields due to the similarity of their defined syntax to that in Structured Fields. Section 3 lists mapped fields -- those whose syntax needs to be transformed into an underlying data model which is then mapped into that defined by Structured Fields.

While implementations can parse and serialise compatible fields as Structured Fields subject to the caveats in Section 2, a sender cannot generate mapped fields from Section 3 and expect them to be understood and acted upon by the recipient without prior negotiation. This specification does not define such a mechanism.

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.

2. Compatible Fields

The HTTP fields listed in Table 1 can usually have their values handled as Structured Fields according to the listed parsing and serialisation algorithms in [STRUCTURED-FIELDS], subject to the listed caveats.

The listed types are chosen for compatibility with the defined syntax of the field as well as with actual internet traffic. However, not all instances of these fields will successfully parse. This might be because the field value is clearly invalid, or it might be because it is valid but not parseable as a Structured Field.

An application using this specification will need to consider how to handle such field values. Depending on its requirements, it might be advisable to reject such values, treat them as opaque strings, or attempt to recover a structured value from them in an ad hoc fashion.

Field Name	Structured Type
Accept	List
Accept-Encoding	List
Accept-Language	List
Accept-Patch	List
Accept-Post	List
Accept-Ranges	List
Access-Control-Allow-Credentials	Item
Access-Control-Allow-Headers	List

Access-Control-Allow-Methods	List
Access-Control-Allow-Origin	Item
Access-Control-Expose-Headers	List
Access-Control-Max-Age	Item
Access-Control-Request-Headers	List
Access-Control-Request-Method	Item
Age	Item
Allow	List
ALPN	List

Alt-Svc	Dictionary
Alt-Used	Item
Cache-Control	Dictionary
CDN-Loop	List
Clear-Site-Data	List
Connection	List
Content-Encoding	List
Content-Language	List
Content-Length	List
Content-Type	Item
Cross-Origin-Resource-Policy	Item

Expect	Dictionary
Expect-CT	Dictionary
Forwarded	Dictionary
Host	Item
Keep-Alive	Dictionary
Max-Forwards	Item
Origin	Item
Pragma	Dictionary
Prefer	Dictionary
Preference-Applied	Dictionary
Retry-After	Item
Sec-WebSocket-Extensions	List
Sec-WebSocket-Protocol	List

Sec-WebSocket-Version	Item
Server-Timing	List
Surrogate-Control	Dictionary
TE	List
Timing-Allow-Origin	List
Trailer	List
Transfer-Encoding	List
Vary	List

X-Content-Type-Options	Item
X-Frame-Options	Item
X-XSS-Protection	List

Table 1

Note the following caveats regarding compatibility:

Parameter and Dictionary keys: HTTP parameter names are case-insensitive (per Section 5.6.6 of [HTTP]), but Structured Fields require them to be all-lowercase. Although the vast majority of parameters seen in typical traffic are all-lowercase, compatibility can be improved by force-lowercasing parameters when encountered. Likewise, many Dictionary-based fields (e.g., Cache-Control, Expect-CT, Pragma, Prefer, Preference-Applied, Surrogate-Control) have case-insensitive keys, and compatibility can be improved by force-lowercasing them.

Parameter delimitation: The parameters rule in HTTP (see Section 5.6.6 of [HTTP]) allows whitespace before the ";" delimiter, but Structured Fields does not. Compatibility can be improved by allowing such whitespace.

String quoting: Section 5.6.4 of [HTTP] allows backslash-escaping most characters in quoted strings, whereas Structured Field Strings only escapes "" and DQUOTE. Compatibility can be improved by unescaping other characters before processing as Strings.

Token limitations: In Structured Fields, tokens are required to

begin with an alphabetic character or "*", whereas HTTP tokens allow a wider range of characters. This prevents use of mapped values that begin with one of these characters. For example, media types, field names, methods, range-units, character and transfer codings that begin with a number or special character other than "*" might be valid HTTP protocol elements, but will not be able to be parsed as Structured Field Tokens.

Integer limitations: Structured Fields Integers can have at most 15 digits; larger values will not be able to be represented in them.

IPv6 Literals: Fields whose values can contain IPv6 literal addresses (such as CDN-Loop, Host, and Origin) are not compatible when those values are parsed as Structured Fields Tokens, because the brackets used to delimit them are not allowed in Tokens.

Empty Field Values: Empty and whitespace-only field values are considered errors in Structured Fields. For compatible fields, an empty field indicates that the field should be silently ignored.

Alt-Svc: Some ALPN tokens (e.g., h3-Q43) do not conform to key's syntax. Since the final version of HTTP/3 uses the h3 token, this shouldn't be a long-term issue, although future tokens may again violate this assumption.

Content-Length: Content-Length is defined as a List because it is not uncommon for implementations to mistakenly send multiple values. See Section 8.6 of [HTTP] for handling requirements.

Retry-After: Only the delta-seconds form of Retry-After is supported; a Retry-After value containing a http-date will need to be either converted into delta-seconds or represented as a raw value.

3. Mapped Fields

Some HTTP fields can have their values represented in Structured Fields by mapping them into its data types and then serialising the result using an alternative field name.

For example, the Date HTTP header field carries a string representing a date:

Date: Sun, 06 Nov 1994 08:49:37 GMT

Its value is more efficiently represented as an integer number of delta seconds from the Unix epoch (00:00:00 UTC on 1 January 1970, minus leap seconds). Thus, the example above would be mapped as:

SF-Date: 784072177

As in Section 2, these fields are unable to represent values that are not parseable, and so an application using this specification will need to how to support such values. Typically, handling them using the original field name is sufficient.

Each field name listed below indicates a replacement field name and a means of mapping its original value into a Structured Field.

3.1. URLs

The field names in Table 2 (paired with their mapped field names) have values that can be represented as Structured Fields by considering the original field's value as a string.

Field Name	Mapped Field Name
Content-Location	SF-Content-Location
Location	SF-Location
Referer	SF-Referer

Table 2

For example, a Location field could be represented as:

SF-Location: "https://example.com/foo"

3.2. Dates

The field names in Table 3 (paired with their mapped field names) have values that can be represented as Structured Fields by parsing their payload according to Section 5.6.7 of [HTTP] and representing the result as an integer number of seconds delta from the Unix Epoch (00:00:00 UTC on 1 January 1970, minus leap seconds).

Field Name	Mapped Field Name
Date	SF-Date
Expires	SF-Expires
If-Modified-Since	SF-IMS
If-Unmodified-Since	SF-IUS
Last-Modified	SF-LM

Table 3

For example, an Expires field could be represented as:

SF-Expires: 1571965240

3.3. ETags

The field value of the ETag header field can be represented as a String Structured Field by representing the entity-tag as a string, and the weakness flag as a boolean "w" parameter on it, where true indicates that the entity-tag is weak; if 0 or unset, the entity-tag is strong.

For example:

SF-ETag: "abcdef"; w=?1

If-None-Match's field value can be represented as SF-INM, which is a List of the structure described above.

For example:

SF-INM: "abcdef"; w=?1, "ghijkl"

3.4. Links

The field value of the Link header field [RFC8288] can be represented in the SF-Link List Structured Field by representing the URI-Reference as a string, and link-param as parameters.

For example:

```
SF-Link: "/terms"; rel="copyright"; anchor="#foo"
```


3.5. Cookies

The field values of the Cookie and Set-Cookie fields [RFC6265] can be represented in the SF-Cookie Structured Field (a List) and SF-Set-Cookie Structured Field (a Dictionary), respectively.

In each case, cookie names are serialized as tokens, whereas their values are serialised as Strings, unless they can be represented accurately and unambiguously using the textual representation of another structured types (e.g., an Integer or Decimal).

Set-Cookie parameters map to parameters on the appropriate SF-Set-Cookie member, with the parameter name being forced to lowercase. Set-Cookie parameter values are Strings unless a specific type is defined. This specification defines the parameter types in Table 4.

Parameter Name	Structured Type
Max-Age	Integer
Secure	Boolean
HttpOnly	Boolean
SameSite	Token

Table 4

Note that cookies in both fields are separated by commas, not semicolons, and multiple cookies can appear in each field.

For example:

```
SF-Set-Cookie: lang=en-US; expires="Wed, 09 Jun 2021 10:18:14 GMT";
               samesite=Strict
SF-Cookie: SID=31d4d96e407aad42, lang=en-US
```

4. IANA Considerations

Please add the following note to the "Hypertext Transfer Protocol (HTTP) Field Name Registry":

> The "Structured Type" column indicates the type of the field (per RFC8941), if any, and may be "Dictionary", "List" or "Item". A prefix of "*" indicates that it is a retrofit type (i.e., not natively Structured); see [this specification].

> Note that field names beginning with characters other than ALPHA or "*" will not be able to be represented as a Structured Fields Token, and therefore may be incompatible with being mapped into fields that refer to it; see [this specification].

Then, add a new column, "Structured Type", with the values from Section 2 assigned to the nominated registrations, prefixing each with "*" to indicate that it is a retrofit type.

Then, add the field names in Table 5, with the corresponding Structured Type as indicated, a status of "permanent" and referring to this document.

Field Name	Structured Type
SF-Content-Location	String
SF-Location	String
SF-Referer	String
SF-Date	Item
SF-Expires	Item
SF-IMS	Item
SF-IUS	Item
SF-LM	Item

SF-ETag	Item
SF-INM	List
SF-Link	List
SF-Set-Cookie	Dictionary
SF-Cookie	List

Table 5

Finally, add the indicated structured type for each existing registry entry below:

Field Name	Structured Type
Accept-CH	List
Cache-Status	List
CDN-Cache-Control	Dictionary
Cross-Origin-Opener-Policy	Item
Cross-Origin-Opener-Policy-Report-Only	Item
Cross-Origin-Embedder-Policy	Item
Cross-Origin-Embedder-Policy-Report-Only	Item
Origin-Agent-Cluster	Item
Priority	Dictionary
Proxy-Status	List

Table 6

5. Security Considerations

Section 2 identifies existing HTTP fields that can be parsed and serialised with the algorithms defined in [STRUCTURED-FIELDS]. Variances from other implementations might be exploitable, particularly if they allow an attacker to target one implementation in a chain (e.g., an intermediary). However, given the considerable variance in parsers already deployed, convergence towards a single parsing algorithm is likely to have a net security benefit in the longer term.

Section 3 defines alternative representations of existing fields. Because downstream consumers might interpret the message differently based upon whether they recognise the alternative representation, implementations are prohibited from generating such fields unless they have negotiated support for them with their peer. This specification does not define such a mechanism, but any such definition needs to consider the implications of doing so carefully.

6. Normative References

[HTTP] Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantics-19, 12 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-semantics-19>.

[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/rfc/rfc2119>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <https://www.rfc-editor.org/rfc/rfc6265>.

[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/rfc/rfc8174>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/rfc/rfc8288>.

[STRUCTURED-FIELDS] Nottingham, M. and P-H. Kamp, "Structured Field Values for HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021, <https://www.rfc-editor.org/rfc/rfc8941>.

Author's Address

Mark Nottingham
Prahran
Australia
Email: mnot@mnot.net
URI: https://www.mnot.net/