Workgroup: Network Working Group
Internet-Draft:
draft-nottingham-http-structure-retrofit-01
Published: 6 January 2022
Intended Status: Informational
Expires: 10 July 2022
Authors: M. Nottingham

Retrofit Structured Fields for HTTP

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-nottingham-http-structure-retrofit/.

information can be found at https://mnot.github.io/I-D/.

Source for this draft and an issue tracker can be found at https://github.com/mnot/I-D/labels/http-structure-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 10 July 2022.

Table of Contents

1. Introduction

Structured Field Values for HTTP [STRUCTURED-FIELDS] introduced a data model with associated parsing and serialisation 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

HTTP fields with the following names 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 (see Appendix A). 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.

*Accept - List

*Accept-Encoding - List

*Accept-Language - List

*Accept-Patch - 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

*Connection - List

*Content-Encoding - List

*Content-Language - List

*Content-Length - List

*Content-Type - Item

*Cross-Origin-Resource-Policy - Item

*Expect - Item

*Expect-CT - Dictionary

*Host - Item

*Keep-Alive - Dictionary

*Origin - Item

*Pragma - Dictionary

*Prefer - Dictionary

*Preference-Applied - Dictionary

*Retry-After - Item

*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

Note the following caveats:

Parameter names: HTTP parameter names are case-insensitive (as 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.

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.

Cache-Control, Expect-CT, Pragma, Prefer, Preference-Applied, Surrogate-Control:
These Dictionary-based fields consider the key to be case-insensitive, but Structured Fields requires keys to be all-lowercase. Although the vast majority of values seen in typical traffic are all-lowercase, compatibility can be improved by force-lowercasing these Dictionary keys when encountered.

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 following field names (paired with their replacement field names) have values that can be represented as Structured Fields by considering the original field's value as a string.

- Content-Location - SF-Content-Location
- Location - SF-Location
- Referer - SF-Referer

For example, a Location field could be represented as:

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

3.2. Dates

The following field names (paired with their replacement field names) have values that can be represented as Structured Fields by parsing their payload according to Section 7.1.1.1 of [RFC7231] 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).

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

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 following parameter types:

- *Max-Age: Integer
- *Secure: Boolean
- *HttpOnly: Boolean
- *SameSite: Token

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 HTTP Field Name Registry:

> The "Structured Type" column indicates the type of the field as 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].

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 following field names into the HTTP Field Name Registry, with the corresponding Structured Type as indicated, a status of "permanent" and referring to this document:

- *SF-Content-Location - String
- *SF-Location - String
- *SF-Referer - String
- *SF-Date - Integer
- *SF-Expires - Integer
- *SF-IMS - Integer

*SF-IUS - Integer

*SF-LM - Integer

*SF-ETag - Item

*SF-INM - List

*SF-Link - List

*SF-Set-Cookie - Dictionary

*SF-Cookie - List

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>.

[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/rfc/rfc7231>.

[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>.

Appendix A. Data Supporting Field Compatibility

To help guide decisions about compatible fields, the HTTP response headers captured by the HTTP Archive https://httparchive.org in September 2021 (representing more than 528,000,000 HTTP exchanges) were parsed as Structured Fields using the types listed in Section 2, with the indicated number of successful header instances, failures, and the resulting failure rate:

accept	9,099	/	34	= 0.372%*
accept-encoding	116,708	/	58	= 0.050%*
accept-language	127,710	/	95	= 0.074%*
accept-patch	281	/	0	= 0.000%
accept-ranges	289,341,375	/	7,776	= 0.003%
access-control-allow-credentials	36,159,371	/	2,671	= 0.007%
access-control-allow-headers	25,980,519	/	23,181	= 0.089%
access-control-allow-methods	32,071,437	/	17,424	= 0.054%
access-control-allow-origin	165,719,859	/	130,247	= 0.079%
access-control-expose-headers	20,787,683	/	1,973	= 0.009%
access-control-max-age	9,549,494	/	9,846	= 0.103%
access-control-request-headers	165,882	/	503	= 0.302%*
access-control-request-method	346,135	/	30,680	= 8.142%*
age	107,395,872	/	36,649	= 0.034%
allow	579,822	/	281	= 0.048%
alt-svc	56,773,977	/	4,914,119	= 7.966%
cache-control	395,402,834	/	1,146,080	= 0.289%
connection	112,017,641	/	3,491	= 0.003%
content-encoding	225,568,224	/	237	= 0.000%
content-language	3,339,291	/	1,744	= 0.052%
content-length	422,415,406	/	126	= 0.000%
content-type	503,950,894	/	507,133	= 0.101%
cross-origin-resource-policy	102,483,430	/	799	= 0.001%
expect	0	/	53	= 100.000%*
expect-ct	54,129,244	/	80,333	= 0.148%
host	57,134	/	1,486	= 2.535%*
keep-alive	50,606,877	/	1,509	= 0.003%
origin	32,438	/	1,396	= 4.126%*
pragma	66,321,848	/	97,328	= 0.147%
preference-applied	189	/	0	= 0.000%
referrer-policy	14,274,787	/	8,091	= 0.057%
retry-after	523,533	/	7,585	= 1.428%
surrogate-control	282,846	/	976	= 0.344%
te	1	/	0	= 0.000%
timing-allow-origin	91,979,983	/	8	= 0.000%
trailer	1,171	/	0	= 0.000%
transfer-encoding	15,098,518	/	0	= 0.000%
vary	246,483,644	/	69,607	= 0.028%
x-content-type-options	166,063,072	/	237,255	= 0.143%
x-frame-options	56,863,322	/	1,014,464	= 1.753%
x-xss-protection	132,739,109	/	347,133	= 0.261%

Note that this data set only includes response headers, although some request headers are present, indicated with an asterisk (because, the Web). Also, Dictionary and Parameter keys have not been force-lowercased, with the result that any values containing uppercase keys are considered to fail.

The top thirty header fields in that data set that were not considered compatible are (* indicates that the field is mapped in Section 3):

**date: 524,810,577

*server: 470,777,294

**last-modified: 383,437,099

**expires: 292,109,781

**etag: 255,788,799

*strict-transport-security: 111,993,787

*x-cache: 70,713,258

*via: 55,983,914

*cf-ray: 54,556,881

*p3p: 54,479,183

*report-to: 54,056,804

*cf-cache-status: 53,536,789

*nel: 44,815,769

*x-powered-by: 37,281,354

*content-security-policy-report-only: 33,104,387

**location: 30,533,957

*x-amz-cf-pop: 28,549,182

*x-amz-cf-id: 28,444,359

*content-security-policy: 25,404,401

*x-served-by: 23,277,252

*x-cache-hits: 21,842,899

**link: 20,761,372

*x-timer: 18,780,130

*content-disposition: 18,516,671

*x-request-id: 16,048,668

*referrer-policy: 15,596,734

*x-cdn: 10,153,756

*x-amz-version-id: 9,786,024

*x-amz-request-id: 9,680,689

*x-dc: 9,557,728

Author's Address

Mark Nottingham
Prahran
VIC
Australia

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