Workgroup: Network Working Group
Internet-Draft: draft-ietf-httpbis-retrofit-05
Updates: 8941 (if approved)
Published: 4 December 2022
Intended Status: Standards Track
Expires: 7 June 2023
Authors: M. Nottingham

Retrofit Structured Fields for HTTP

Abstract

This specification nominates a selection of existing HTTP fields as having syntax that is compatible with Structured Fields, so that they can be handled as such (subject to certain caveats).

To accommodate some additional fields whose syntax is not compatible, it also defines mappings of their semantics into new Structured Fields. It does not specify how to negotiate their use.

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 will expire on 7 June 2023.

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. 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.

Note that 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
DNT	Item
Expect	Dictionary
Expect-CT	Dictionary
Host	Item
Keep-Alive	Dictionary
Max-Forwards	Item
Origin	Item
Pragma	Dictionary

Field Name	Structured Type
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
Upgrade-Insecure-Requests	Item
Vary	List
X-Content-Type-Options	Item
X-Frame-Options	Item
X-XSS-Protection	List

Table 1: Compatible Fields

Note the following caveats regarding compatibility:

Error handling: Parsing algorithms specified (or just widely implemented) for current HTTP headers may differ from those in Structured Fields in details such as error handling. For example, HTTP specifies that repeated directives in the Cache-Control header field have a different precedence than that assigned by a Dictionary structured field (which Cache-Control is mapped to).

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 parsing. 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 when parsing.

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 when parsing.

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

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 represented 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 contain IPv6 literal addresses (such as CDN-Loop, Host, and Origin) are not able to be represented 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, and therefore cannot be represented as a Token. 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: Note that 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 can be represented; a Retry-After value containing a http-date will need to be converted into delta-seconds to be conveyed as a Structured Field Value.

3. Mapped Fields

Some HTTP field values have syntax that cannot be successfully parsed as Structured Fields. Instead, it is necessary to map them into a separate Structured Field with an alternative name.

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

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

Its value would be mapped to:

```
SF-Date: @784111777
```

As in Section 2, these fields are unable to carry values that are not valid Structured Fields, 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 mapped into Structured Fields by treating 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: URL Fields

For example, this Location field

```
Location: https://example.com/foo
```

could be mapped 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 mapped into Structured Fields by parsing their payload according to Section 5.6.7 of [HTTP] and representing the result as a Date.

Field Name	Mapped Field Name
Date	SF-Date
Expires	SF-Expires

Field Name	Mapped Field Name
If-Modified-Since	SF-If-Modified-Since
If-Unmodified-Since	SF-If-Unmodified-Since
Last-Modified	SF-Last-Modified

Table 3: Date Fields

For example, an Expires field could be mapped as:

```
SF-Expires: @1659578233
```

3.3. ETags

The field value of the ETag header field can be mapped into the SF-ETag 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, this:

```
ETag: W/"abcdef"
```

```
SF-ETag: "abcdef"; w
```

If-None-Match's field value can be mapped into the SF-If-None-Match Structured Field, which is a List of the structure described above. When a field value contains "*", it is represented as a Token.

Likewise, If-Match's field value can be mapped into the SF-If-Match Structured Field in the same manner.

For example:

```
SF-If-None-Match: "abcdef"; w, "ghijkl", *
```

3.4. Links

The field value of the Link header field [RFC8288] can be mapped into the SF-Link List Structured Field by considering the URI-Reference as a String, and link-param as Parameters.

For example, this:

```
Link: </terms>; rel="copyright"; anchor="#foo"
```

can be mapped to:

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

3.5. Cookies

The field values of the Cookie and Set-Cookie fields [COOKIES] can be mapped into the SF-Cookie Structured Field (a List) and SF-Set-Cookie Structured Field (a List), respectively.

In each case, a cookie is represented as an Inner List containing two Items; the cookie name and value. The cookie name is always a String; the cookie value is a String, unless it can be successfully parsed as the textual representation of another, bare Item structured type (e.g., Byte Sequence, Decimal, Integer, Token, or Boolean).

Cookie attributes map to Parameters on the Inner List, with the parameter name being forced to lowercase. Cookie attribute values are Strings unless a specific type is defined for them. This specification defines types for existing cookie attributes in Table 4.

Parameter Name	Structured Type
Domain	String
HttpOnly	Boolean
Expires	Date
Max-Age	Integer
Path	String
Secure	Boolean
SameSite	Token

Table 4: Set-Cookie Parameter Types

The Expires attribute is mapped to a Date representation of parsed-cookie-date (see Section 5.1.1 of [COOKIES]).

For example, these unstructured fields:

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

can be mapped into:

```
SF-Set-Cookie: ("lang" "en-US"); expires=@1623233894;
               samesite=Strict; secure
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	Item
SF-Cookie	List
SF-Date	Item
SF-ETag	Item
SF-Expires	Item
SF-If-Match	List
SF-If-Modified-Since	Item
SF-If-None-Match	List
SF-If-Unmodified-Since	Item
SF-Link	List
SF-Last-Modified	Item
SF-Location	Item
SF-Referer	Item
SF-Set-Cookie	List

Table 5: New Fields

Then, add the indicated Structured Type for each existing registry entry listed in Table 6.

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

Table 6: Existing Fields

Finally, add a new column to the "Cookie Attribute Registry" established by [COOKIES] with the title "Structured Type", using information from Table 4.

5. Security Considerations

Section 2 identifies existing HTTP fields that can be parsed and serialised with the algorithms defined in [STRUCTURED-FIELDS]. Variances from existing parser behavior 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

[COOKIES] Bingler, S., West, M., and J. Wilander, "Cookies: HTTP State Management Mechanism", Work in Progress, Internet-Draft, draft-ietf-httpbis-rfc6265bis-11, 7 November 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-rfc6265bis-11>.

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[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>.

[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. Kamp, "Structured Field Values for HTTP", Work in Progress, Internet-Draft, draft-ietf-httpbis-sfbis-00, 9 November 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-sfbis-00>.

Author's Address

Mark Nottingham
Prahran
Australia

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