Workgroup: Network Working Group
Internet-Draft: draft-nottingham-link-hint-03
Published: 23 July 2023
Intended Status: Informational
Expires: 24 January 2024
Authors: M. Nottingham

HTTP Link Hints

Abstract

This memo specifies "HTTP Link Hints", a mechanism for annotating Web links to HTTP(S) resources with information that otherwise might be discovered by interacting with them.

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-link-hint/.

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/link-hint.

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 24 January 2024.

Table of Contents

1. Introduction

HTTP [RFC7230] clients can discover a variety of information about a resource by interacting with it. For example, the methods supported can be learned through the Allow response header field, and the need for authentication is conveyed with a 401 Authentication Required status code.

Often, it can be beneficial to know this information before interacting with the resource; not only can such knowledge save time (through reduced round trips), but it can also affect the choices available to the code or user driving the interaction.

For example, a user interface that presents the data from an HTTP-based API might need to know which resources the user has write access to, so that it can present the appropriate interface.

This specification defines a vocabulary of "HTTP link hints" that allow such metadata about HTTP resources to be attached to Web links [RFC8288], thereby making it available before the link is followed. It also establishes a registry for future hints.

Hints are just that -- they are not a "contract", and are to only be taken as advisory. The runtime behaviour of the resource always overrides hinted information.

For example, a client might receive a hint that the PUT method is allowed on all "widget" resources. This means that generally, the client can PUT to them, but a specific resource might reject a PUT based upon access control or other considerations.

More fine-grained information might also be gathered by interacting with the resource (e.g., via a GET), or by another resource "containing" it (such as a "widgets" collection) or describing it (e.g., one linked to it with a "describedby" link relation).

There is not a single way to carry hints in a link; rather, it is expected that this will be done by individual link serialisations (see [RFC8288], Section 3.4.1). However, Appendix A does recommend how to include link hints in the existing Link HTTP header field.

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. HTTP Link Hints

A HTTP link hint is a (key, value) tuple that describes the target resource of a Web link [RFC8288], or the link itself. The value's canonical form is a JSON [RFC8259] data structure specific to that hint.

Typically, link hints are serialised in links as target attributes ([RFC8288], Section 3.4.1).

In JSON-based formats, this can be achieved by simply serialising link hints as an object; for example:

```
{
  "_links": {
    "self": {
      "href": "/orders/523",
       "hints": {
        "allow": ["GET", "POST"],
        "accept-post": {
          "application/example+json":
            {}
        }
      }
    }
  }
}
```

In other link formats, this requires a mapping from the canonical JSON data model. One such mapping is described in Appendix A for the Link HTTP header field.

The information in a link hint SHOULD NOT be considered valid for longer than the freshness lifetime ([RFC7234], Section 4.2) of the representation that the link occurred within, and in some cases, it might be valid for a considerably shorter period.

Likewise, the information in a link hint is specific to the link it is attached to. This means that if a representation is specific to a particular user, the hints on links in that representation are also specific to that user.

3. Pre-Defined HTTP Link Hints

3.1. allow

*Hint Name: allow

*Description: Hints the HTTP methods that can be used to interact with the target resource; equivalent to the Allow HTTP response header.

*Content Model: array (of strings)

*Specification: [this document]

Content MUST be an array of strings, containing HTTP methods ([RFC7231], Section 4).

3.2. formats

*Hint Name: formats

*Description: Hints the representation type(s) that the target resource can produce and consume, using the GET and PUT (if allowed) methods respectively.

*Content Model: object

*Specification: [this document]

Content MUST be an object, whose keys are media types ([RFC7231], Section 3.1.1.1), and values are objects.

The object MAY have a "links" member, whose value is an object representing links (in the sense of [RFC8288]) whose context is any document that uses that format. Generally, this will be schema or profile ([RFC6906]) information. The "links" member has the same format as the "links" hint.

Furthermore, the object MAY have a "deprecated" member, whose value is either true or false, indicating whether support for the format might be removed in the near future.

All other members of the object are under control of the corresponding media type's definition.

3.3. links

*Hint Name: links

*Description: Hints at links whose context is the target resource.

*Content Model: object

*Specification: [this document]

The "links" hint contains links (in the sense of [RFC8288]) whose context is the hinted target resource, which are stable for the lifetime of the hint.

Content MUST be an object, whose member names are link relations ([RFC8288]) and values are objects that MUST have an "href" member whose value is a URI-reference ([RFC3986], using the original link as the base for resolution) for the link hint's target resource, and MAY itself contain link hints, serialised as the value for a "hints" member.

For example:

```
"links": {
  "edit-form": {
    "href": "./edit",
    "hints": {
      "formats": {
        "application/json": {}
      }
    }
  }
}
```

3.4. accept-post

*Hint Name: accept-post

*Description: Hints the POST request format(s) that the target resource can consume.

*Content Model: object

*Specification: [this document]

Content MUST be an object, with the same constraints as for "formats".

When this hint is present, "POST" SHOULD be listed in the "allow" hint.

3.5. accept-patch

*Hint Name: accept-patch

*Description: Hints the PATCH [RFC5789] request format(s) that the target resource can consume; equivalent to the Accept-Patch HTTP response header.

*Content Model: array (of strings)

*Specification: [this document]

Content MUST be an array of strings, containing media types ([RFC7231], Section 3.1.1.1).

When this hint is present, "PATCH" SHOULD be listed in the "allow" hint.

3.6. accept-ranges

*Hint Name: accept-ranges

*Description: Hints the range-specifier(s) available for the target resource; equivalent to the Accept-Ranges HTTP response header [RFC7233].

*Content Model: array (of strings)

*Specification: [this document]

Content MUST be an array of strings, containing HTTP range-specifiers ([RFC7233], Section 3.1).

3.7. accept-prefer

*Hint Name: accept-prefer

*Description: Hints the preference(s) [RFC7240] that the target resource understands (and might act upon) in requests.

*Content Model: array (of strings)

*Specification: [this document]

Content MUST be an array of strings, contain preferences ([RFC7240], Section 2). Note that, by its nature, a preference can be ignored by the server.

3.8. precondition-req

*Hint Name: precondition-req

*Description: Hints that the target resource requires state-changing requests (e.g., PUT, PATCH) to include a precondition, as per [RFC7232], to avoid conflicts due to concurrent updates.

*Content Model: array (of strings)

*Specification: [this document]

Content MUST be an array of strings, with possible values "etag" and "last-modified" indicating type of precondition expected.

See also the 428 Precondition Required status code ([RFC6585]).

3.9. auth-schemes

*Hint Name: auth-schemes

*Description: Hints that the target resource requires authentication using the HTTP Authentication Framework [RFC7235].

*Content Model: array (of objects)

*Specification: [this document]

Content MUST be an array of objects, each with a "scheme" member containing a string that corresponds to a HTTP authentication scheme ([RFC7235]), and optionally a "realms" member containing an array of zero to many strings that identify protection spaces that the resource is a member of.

For example:

```
{
  "auth-req": [
    {
      "scheme": "Basic",
      "realms": ["private"]
    }
  ]
}
```

3.10. status

*Hint Name: status

*Description: Hints the status of the target resource.

*Content Model: string

*Specification: [this document]

Content MUST be a string; possible values are:

*"deprecated" - indicates that use of the resource is not recommended, but it is still available.

*"gone" - indicates that the resource is no longer available; i.e., it will return a 410 Gone HTTP status code if accessed.

4. Security Considerations

Clients need to exercise care when using hints. For example, a naive client might send credentials to a server that uses the auth-req hint, without checking to see if those credentials are appropriate for that server.

5. IANA Considerations

5.1. HTTP Link Hint Registry

This specification defines the HTTP Link Hint Registry. See Section 2 for a general description of the function of link hints.

Link hints are generic; that is, they are potentially applicable to any HTTP resource, not specific to one application of HTTP, nor to one particular format. Generally, they ought to be information that would otherwise be discoverable by interacting with the resource.

Hint names MUST be composed of the lowercase letters (a-z), digits (0-9), underscores ("_") and hyphens ("-"), and MUST begin with a lowercase letter.

Hint content MUST be described in terms of JSON values ([RFC8259], Section 3).

Hint semantics SHOULD be described in terms of the framework defined in [RFC8288].

New hints are registered using the Expert Review process described in [RFC8126] to enforce the criteria above. Requests for registration of new resource hints are to use the following template:

*Hint Name: [hint name]

*Description: [a short description of the hint's semantics]

*Content Model: [valid JSON value types; see RFC627 Section 2.1]

*Specification: [reference to specification document]

Initial registrations are enumerated in Section 3. The "rel", "rev", "hreflang", "media", "title", and "type" hint names are reserved, so as to avoid potential clashes with link serialisations.

6. References

6.1. 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/rfc/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, DOI 10.17487/RFC5789, March 2010, <https://www.rfc-editor.org/rfc/rfc5789>.

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, DOI 10.17487/RFC6585, April 2012, <https://www.rfc-editor.org/rfc/rfc6585>.

[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/rfc/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/rfc/rfc7231>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <https://www.rfc-editor.org/rfc/rfc7232>.

[RFC7233] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, DOI 10.17487/RFC7233, June 2014, <https://www.rfc-editor.org/rfc/rfc7233>.

[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/rfc/rfc7234>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <https://www.rfc-editor.org/rfc/rfc7235>.

[RFC7240] Snell, J., "Prefer Header for HTTP", RFC 7240, DOI 10.17487/RFC7240, June 2014, <https://www.rfc-editor.org/rfc/rfc7240>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/

RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

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

6.2. Informative References

[RFC6906] Wilde, E., "The 'profile' Link Relation Type", RFC 6906, DOI 10.17487/RFC6906, March 2013, <https://www.rfc-editor.org/rfc/rfc6906>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

Appendix A. Representing Link Hints in Link Headers

A link hint can be represented in a Link header ([RFC8288], Section 3) as a link-extension.

When doing so, the JSON of the hint's content SHOULD be normalised to reduce extraneous spaces (%x20), and MUST NOT contain horizontal tabs (%x09), line feeds (%x0A) or carriage returns (%x0D). When they are part of a string value, these characters MUST be escaped as described in [RFC8259] Section 7; otherwise, they MUST be discarded.

Furthermore, if the content is an array or an object, the surrounding delimiters MUST be removed before serialisation. In other words, the outermost object or array is represented without the braces ("{}") or brackets ("[]") respectively, but this does not apply to inner objects or arrays.

For example, the two JSON values below are those of the fictitious "example" and "example1" hints, respectively:

```
"The Example Value"
1.2
```

In a Link header, they would be serialised as:

```
Link: </>; rel="sample"; example="The Example Value";
      example1=1.2
```

A more complex, single value for "example":

```
[
  "foo",
  -1.23,
  true,
  ["charlie", "bennet"],
  {"cat": "thor"},
  false
]
```

would be serialised as:

```
Link: </>; rel="sample"; example="\"foo\", -1.23, true,
      [\"charlie\", \"bennet\"], {"cat": \"thor\"}, false"
```

Appendix B. Acknowledgements

Thanks to Jan Algermissen, Mike Amundsen, Bill Burke, Graham Klyne, Leif Hedstrom, Jeni Tennison, Erik Wilde and Jorge Williams for their suggestions and feedback.

Author's Address

Mark Nottingham

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