Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 11 February 2026

M. Nottingham
10 August 2025

HTTP Link Hints
draft-ietf-httpapi-link-hint-04

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.

The latest revision of this draft can be found at https://ietf-wg-httpapi.github.io/link-hint/draft-ietf-httpapi-link-hint.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-httpapi-link-hint/.

Discussion of this document takes place on the Building Blocks for HTTP APIs Working Group mailing list (mailto:httpapi@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/httpapi/. Subscribe at https://www.ietf.org/mailman/listinfo/httpapi/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-httpapi/link-lint.

Status of This Memo

Table of Contents

1. Introduction

HTTP [HTTP] clients can discover a variety of information about a resource by interacting with it. For example, the methods supported by a resource can be learned by examining the Allow header field in responses from it, 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 influence the choices made by 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 [WEB-LINKING], 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 send a 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 convey hints with a link; rather, it is expected that this will be done by individual link serialisations (see Section 3.4.1 of [WEB-LINKING]).

Finally, these hints are not universally applicable to all links. Section 2.2 of [WEB-LINKING] specifies that the name space of target attributes for a given link is defined by both the link relation in use, and by the serialisation in use. Therefore, to be used in a given link, these hints need to be explicitly specified by either the link relation type or the serialisation (e.g., by referencing this specification).

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 [WEB-LINKING], or describes the link itself. The value's canonical form is expressed in subset of the data types defined by HTTP Structured Fields [STRUCTURED-FIELDS].

Typically, link hints are serialised in links as target attributes (Section 3.4.1 of [WEB-LINKING]). In the Link HTTP Header, this can be done by serialising those attributes as strings. In other link formats, this requires a mapping from the canonical data model into the constraints of that format.

The information in a link hint SHOULD NOT be considered valid for longer than the freshness lifetime (Section 4.2 of [HTTP-CACHING]) 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: Inner List of Strings

* Specification: [this document]

Content MUST be an Inner List of Strings, containing HTTP methods (Section 9 of [HTTP]).

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: Inner List of Strings

* Specification: [this document]

Content MUST be an Inner List of Strings, containing media types (Section 8.3.1 of [HTTP]).

3.3. accept-post

* Hint Name: accept-post

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

* Content Model: Inner List of Strings

* Specification: [this document]

Content MUST be an Inner List of Strings, with the same constraints as for "formats".

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

3.4. accept-patch

* Hint Name: accept-patch

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

* Content Model: Inner List of Strings

* Specification: [this document]

Content MUST be an Inner List of Strings, containing media types (Section 8.3.1 of [HTTP]) that identify the acceptable patch formats (see [PATCH]).

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

3.5. accept-query

* Hint name: accept-query

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

* Content Model: Inner List (of Strings)

* Specification: [this document]

Content MUST be an Inner List of Strings, containing media types (Section 8.3.1 of [HTTP]) that identify the acceptable query formats (see [QUERY]).

When this hint is present, "QUERY" SHOLD be listed in the "allow" hint, when it is present.

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 [HTTP].

* Content Model: Inner List of Strings

* Specification: [this document]

Content MUST be an Inner List of Strings, containing HTTP ranges-specifiers (Section 14.1.1 of [HTTP]).

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: Inner List of Strings

* Specification: [this document]

Content MUST be an Inner List of Strings, containing preferences (Section 2 of [RFC7240]). 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 Section 13.1 of [HTTP], to avoid conflicts due to concurrent updates.

* Content Model: Inner List of Strings

* Specification: [this document]

Content MUST be an Inner List 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 Section 11 of [HTTP].

* Content Model: Inner List of Strings

* Specification: [this document]

Content MUST be an Inner List of Strings, each corresponding to a HTTP authentication scheme (Section 11.1 of [HTTP]).

3.10. auth-realms

* Hint Name: auth-realms

* Description: Hints the authentication realm(s) available for those schemes that support them in the HTTP Authentication framework Section 11 of [HTTP].

* Content Model: Inner List (of Strings)

* Specification: [this document]

Content MUST be an Inner List of Strings, each indicating a protection space that the resource is a member of.

3.11. status

* Hint Name: status
* Description: Hints the status of the target resource.
* Content Model: Token
* Specification: [this document]

Content MUST be a Token; 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 using valid combinations of the following types defined by HTTP Structured Fields ([STRUCTURED-FIELDS]):

* Inner List (Section 3.1.2 of [STRUCTURED-FIELDS])
* Item (Section 3.3 of [STRUCTURED-FIELDS])

Hint semantics SHOULD be described in terms of the framework defined in [WEB-LINKING].

New hints are registered using the IETF Review process described in [RFC8126]. 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: [allowed Structured Fields types]

* Specification: [reference to specification document]

Initial registrations are enumerated in Section 3. Additionally, 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

[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>.

[HTTP-CACHING] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", STD 98, RFC 9111, DOI 10.17487/RFC9111, June 2022, <https://www.rfc-editor.org/rfc/rfc9111>.

[JSON] 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>.

[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>.

[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>.

[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>.

[STRUCTURED-FIELDS] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", RFC 9651, DOI 10.17487/RFC9651, September 2024, <https://www.rfc-editor.org/rfc/rfc9651>.

[URI] 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>.

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

6.2. Informative References

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

[QUERY] Reschke, J., Snell, J. M., and M. Bishop, "The HTTP QUERY Method", Work in Progress, Internet-Draft, draft-ietf-httpbis-safe-method-w-body-11, 19 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-safe-method-w-body-11>.

[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. 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/