

Network Working Group
Internet-Draft
Obsoletes: [5988](#) (if approved)
Intended status: Standards Track
Expires: February 12, 2018

M. Nottingham
August 11, 2017

Web Linking

draft-nottingham-rfc5988bis-08

Abstract

This specification defines a model for the relationships between resources on the Web ("links") and the type of those relationships ("link relation types").

It also defines the serialisation of such links in HTTP headers with the Link header field.

Note to Readers

RFC EDITOR: please remove this section before publication

This is a work-in-progress to revise [RFC5988](#).

The issues list can be found at <https://github.com/mnot/I-D/labels/rfc5988bis>.

The most recent (often, unpublished) draft is at <https://mnot.github.io/I-D/rfc5988bis/>.

Recent changes are listed at <https://github.com/mnot/I-D/commits/gh-pages/rfc5988bis>.

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 <http://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 February 12, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](http://trustee.ietf.org/license-info) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	3
1.1.	Notational Conventions	3
1.2.	Conformance and Error Handling	4
2.	Links	4
2.1.	Link Relation Types	5
2.1.1.	Registered Relation Types	5
2.1.2.	Extension Relation Types	7
2.2.	Target Attributes	8
3.	Link Serialisation in HTTP Headers	8
3.1.	Link Target	9
3.2.	Link Context	9
3.3.	Relation Type	10
3.4.	Target Attributes	11
3.4.1.	Serialisation-Defined Attributes	11
3.4.2.	Extension Attributes	12
3.5.	Link Header Field Examples	12
4.	IANA Considerations	14

Nottingham

Expires February 12, 2018

[Page 2]

4.1.	Link HTTP Header Field Registration	14
4.2.	Link Relation Type Registry	14
4.3.	Link Relation Application Data Registry	14
5.	Security Considerations	14
6.	Internationalisation Considerations	15
7.	References	15
7.1.	Normative References	15
7.2.	Informative References	17
Appendix A.	Notes on Other Link Serialisations	18
A.1.	Link Serialisation in HTML	18
A.2.	Link Serialisation in Atom	18
Appendix B.	Algorithms for Parsing Link Header Fields	19
B.1.	Parsing a Header Set for Links	19
B.2.	Parsing a Link Field Value	20
B.3.	Parsing Parameters	22
B.4.	Parsing a Quoted String	23
Appendix C.	Changes from RFC5988	24
	Author's Address	25

[1.](#) Introduction

This specification defines a model for the relationships between resources on the Web ("links") and the type of those relationships ("link relation types").

HTML [[W3C.REC-html5-20141028](#)] and Atom [[RFC4287](#)] both have well-defined concepts of linking; [Section 2](#) generalises this into a framework that encompasses linking in these formats and (potentially) elsewhere.

Furthermore, [Section 3](#) defines an HTTP header field for conveying such links.

[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.

This document uses the Augmented Backus-Naur Form (ABNF) notation of [[RFC7230](#)], including the #rule, and explicitly includes the following rules from it: quoted-string, token, SP (space), BWS (bad whitespace), OWS (optional whitespace), RWS (required whitespace) LOALPHA, DIGIT.

Nottingham

Expires February 12, 2018

[Page 3]

Additionally, the following rules are included from [\[RFC3986\]](#): URI and URI-Reference; from [\[RFC6838\]](#): type-name and subtype-name; from [\[W3C.REC-css3-mediaqueries-20120619\]](#): media-query-list; and from [\[RFC5646\]](#): Language-Tag.

1.2. Conformance and Error Handling

The requirements regarding conformance and error handling highlighted in [\[RFC7230\]](#), [Section 2.5](#) apply to this document.

2. Links

In this specification, a link is a typed connection between two resources, and is comprised of:

- o A `_link context_`,
- o a `_link relation type_` ([Section 2.1](#)),
- o a `_link target_`, and
- o optionally, `_target attributes_` ([Section 2.2](#)).

A link can be viewed as a statement of the form "`_link context_` has a `_link relation type_` resource at `_link target_`, which has `_target attributes_`".

For example, `https://www.example.com/` has a "canonical" resource at `https://example.com`, which has a "type" of "text/html".

Link contexts and link targets are both IRIs [\[RFC3987\]](#). However, in the common case, the link context will also be a URI [\[RFC3986\]](#), because many protocols (such as HTTP) do not support dereferencing IRIs. Likewise, the link target will be sometimes be converted to a URI (see [\[RFC3987\]](#), [Section 3.1](#)) in serialisations that do not support IRIs (such as the Link header field defined in [Section 3](#)).

This specification does not place restrictions on the cardinality of links; there can be multiple links to and from a particular target, and multiple links of the same or different types between a given context and target. Likewise, the relative ordering of links in any particular serialisation, or between serialisations (e.g., the Link header field and in-content links) is not specified or significant in this specification; applications that wish to consider ordering significant can do so.

Links are conveyed in `_link serialisations_`; they are the "bytes on the wire", and can occur in various forms. For example, Atom

Nottingham

Expires February 12, 2018

[Page 4]

[RFC4287] and HTML [[W3C.REC-html5-20141028](#)] both defined serialisations of links into their respective formats, and [Section 3](#) defines how to serialise links in HTTP header fields.

This specification does not define a general syntax for links across different serialisations, nor does it mandate a specific context for any given link; it is expected that serialisations of links will specify both aspects.

Finally, links are used by `_link applications_`. Generally, an application will define the link relation type(s) it uses, along with the serialisation(s) that they might occur within. For example, the application "Web browsing" looks for the "stylesheet" link relation type in the HTML link serialisation (and optionally in the Link header field), whereas the application "AtomPub" uses the "edit" and "edit-media" link relations in the Atom serialisation.

[2.1.](#) Link Relation Types

In the simplest case, a link relation type identifies the semantics of a link. For example, a link with the relation type "copyright" indicates that the current link context has a copyright resource at the link target.

Link relation types can also be used to indicate that the target resource has particular attributes, or exhibits particular behaviours; for example, a "service" link implies that the link target can be used as part of a defined protocol (in this case, a service description).

Relation types are not to be confused with media types [[RFC2046](#)]; they do not identify the format of the representation that results when the link is dereferenced. Rather, they only describe how the current context is related to another resource.

Relation types SHOULD NOT infer any additional semantics based upon the presence or absence of another link relation type, or its own cardinality of occurrence. An exception to this is the combination of the "alternate" and "stylesheet" registered relation types, which has special meaning in HTML for historical reasons.

There are two kinds of relation types: registered and extension.

[2.1.1.](#) Registered Relation Types

Well-defined relation types can be registered as tokens for convenience and/or to promote reuse by other applications, using the procedure in [Section 2.1.1.1](#).

Nottingham

Expires February 12, 2018

[Page 5]

Registered relation type names MUST conform to the reg-rel-type rule (see [Section 3.3](#)), and MUST be compared character-by-character in a case-insensitive fashion. They SHOULD be appropriate to the specificity of the relation type; i.e., if the semantics are highly specific to a particular application, the name should reflect that, so that more general names are available for less specific use.

Registered relation types MUST NOT constrain the media type of the link context, and MUST NOT constrain the available representation media types of the link target. However, they can specify the behaviours and properties of the target resource (e.g., allowable HTTP methods, request and response media types that are required be supported).

Historically, registered relation types have been identified with a URI [[RFC3986](#)] by prefixing their names with an application-defined base URI (e.g., see [Appendix A.2](#)). This practice is NOT RECOMMENDED, because the resulting strings will not be considered equivalent to the registered relation types by other applications. Applications that do use such URIs internally MUST NOT use them in link serialisations that do not explicitly accommodate them.

[2.1.1.1](#). Registering Link Relation Types

The link relations registry is located at <https://www.iana.org/assignments/link-relations/>. Registration requests can be made by following the instructions located there, or by sending an e-mail to the "link-relations@ietf.org" mailing list.

Registration requests consist of at least the following information:

- o *Relation Name*: The name of the relation type
- o *Description*: A short English description of the type's semantics. It SHOULD be stated in terms of the relationship between the link context and link target.
- o *Reference*: Reference to the document that specifies the link relation type, preferably including a URI that can be used to retrieve a copy of the document. An indication of the relevant section(s) can also be included, but is not required.

The expert(s) can define additional fields to be collected in the registry.

General requirements for registered relation types are described in [Section 2.1.1](#).

Nottingham

Expires February 12, 2018

[Page 6]

Registrations MUST reference a freely available, stable specification.

Note that relation types can be registered by third parties (including the expert(s)), if the expert(s) determine that an unregistered relation type is widely deployed and not likely to be registered in a timely manner otherwise. Such registrations still are subject to the requirements defined, including the need to reference a specification.

2.1.1.2. Registration Request Processing

Relation types are registered using the Specification Required policy (see [Section 4.6 of \[RFC8126\]](#)), which implies review and approval by a designated expert.

The goal of the registry is to reflect common use of links on the Internet. Therefore, the expert(s) should be strongly biased towards approving registrations, unless they are abusive, frivolous, not likely to be used on the Internet, or actively harmful to the Internet and/or the Web (not merely aesthetically displeasing, or architecturally dubious). As stated in [Section 2.1.1](#), the expert(s) can withhold registration of names that are too general for the proposed application.

The expert(s) will clearly identify any issues which cause a registration to be refused. Advice about the semantics of a proposed link relation type can be given, but if it does not block registration, this should be explicitly stated.

When a request is approved, the expert(s) will inform IANA, and the registration will be processed. The IESG is the final arbiter of any objection.

2.1.2. Extension Relation Types

Applications that don't wish to register a relation type can use an extension relation type, which is a URI [[RFC3986](#)] that uniquely identifies the relation type. Although the URI can point to a resource that contains a definition of the semantics of the relation type, clients SHOULD NOT automatically access that resource to avoid overburdening its server.

The URI used for an extension relation type SHOULD be under the control of the person or party defining it, or be delegated to them.

When extension relation types are compared, they MUST be compared as strings (after converting to URIs if serialised in a different

Nottingham

Expires February 12, 2018

[Page 7]

format) in a case-insensitive fashion, character-by-character. Because of this, all-lowercase URIs SHOULD be used for extension relations.

Note that while extension relation types are required to be URIs, a serialisation of links can specify that they are expressed in another form, as long as they can be converted to URIs.

2.2. Target Attributes

`_Target attributes_` are a list of key/value pairs that describe the link or its target; for example, a media type hint.

They can be defined both by individual link relation types and by link serialisations.

This specification does not attempt to coordinate the name of target attributes, their cardinality or use. Those creating and maintaining serialisations SHOULD coordinate their target attributes to avoid conflicts in semantics or syntax, and MAY define their own registries of target attributes.

The names of target attributes SHOULD conform to the token rule, but SHOULD NOT include any of the characters "%", "'" or "*", for portability across serializations, and MUST be compared in a case-insensitive fashion.

Target attribute definitions SHOULD specify:

- o The serialisation of their values into Unicode or a subset thereof, to maximise their chances of portability across link serialisations.
- o The semantics and error handling of multiple occurrences of the target attribute on a given link.

This specification does define target attributes for use in the Link HTTP header field in [Section 3.4](#).

3. Link Serialisation in HTTP Headers

The Link header field provides a means for serialising one or more links into HTTP headers.

The ABNF for the field value is:

Nottingham

Expires February 12, 2018

[Page 8]

```
Link           = #link-value
link-value     = "<" URI-Reference ">" *( OWS ";" OWS link-param )
link-param     = token BWS [ "=" BWS ( token / quoted-string ) ]
```

Note that any "link-param" can be generated with values using either the "token" or the "quoted-string" syntax, and therefore recipients MUST be able to parse both forms. In other words, the following parameters are equivalent:

```
x=y
x="y"
```

Previous definitions of the Link header did not equate the token and quoted-string forms explicitly; the "title" parameter was always quoted, and the "hreflang" parameter was always a token. Senders wishing to maximize interoperability will send them in those forms.

Individual "link-param"s specify their syntax in terms of the value after any necessary unquoting (as per [\[RFC7230\], Section 3.2.6](#)).

This specification establishes the link-params "rel", "anchor", and "rev" (which are part of the general link model), as well as "hreflang", "media", "title", "title*", and "type" (which are target attributes defined by the serialization).

[3.1.](#) Link Target

Each link-value conveys one target IRI as a URI-Reference (after conversion to one, if necessary; see [\[RFC3987\], Section 3.1](#)) inside angle brackets ("<>"). If the URI-Reference is relative, parsers MUST resolve it as per [\[RFC3986\], Section 5](#). Note that any base IRI appearing in the message's content is not applied.

[3.2.](#) Link Context

By default, the context of a link conveyed in the Link header field is the URL of the representation it is associated with, as defined in [\[RFC7231\], Section 3.1.4.1](#), serialised as a URI.

When present, the anchor parameter overrides this with another URI, such as a fragment of this resource, or a third resource (i.e., when the anchor value is an absolute URI). If the anchor parameter's value is a relative URI, parsers MUST resolve it as per [\[RFC3986\], Section 5](#). Note that any base URI from the body's content is not applied.

The ABNF for the "anchor" parameter's value is:

Nottingham

Expires February 12, 2018

[Page 9]

URI-Reference ; [Section 4.1](#) of [{{RFC3986}}](#)

Link application can choose to ignore links with an anchor parameter. For example, the application in use might not allow the link context to be assigned to a different resource. In such cases, the entire link is to be ignored; link applications MUST NOT process the link without applying the anchor.

Note that depending on HTTP status code and response headers, the link context might be "anonymous" (i.e., no link context is available). For example, this is the case on a 404 response to a GET request.

[3.3.](#) Relation Type

The relation type of a link conveyed in the Link header field is conveyed in the "rel" parameter's value. The "rel" parameter MUST be present but MUST NOT appear more than once in a given link-value; occurrences after the first MUST be ignored by parsers.

The "rel" parameter can, however, contain multiple link relation types. When this occurs, it establishes multiple links that share the same context, target, and target attributes.

The "rev" parameter has been used in the past to indicate that the semantics of the relationship are in the reverse direction. That is, a link from A to B with REL="X" expresses the same relationship as a link from B to A with REV="X". "rev" is deprecated by this specification because it often confuses authors and readers; in most cases, using a separate relation type is preferable.

The ABNF for the "rel" and "rev" parameters' values is:

```
relation-type *( 1*SP relation-type )
```

where:

```
relation-type  = reg-rel-type / ext-rel-type
reg-rel-type   = LOALPHA *( LOALPHA / DIGIT / "." / "-" )
ext-rel-type   = URI ; Section 3 of {{RFC3986}}
```

Note that extension relation types are REQUIRED to be absolute URIs in Link header fields, and MUST be quoted when they contain characters not allowed in tokens, such as semicolon (";") or comma (",") (as these characters are used as delimiters in the header field itself).

Nottingham

Expires February 12, 2018

[Page 10]

3.4. Target Attributes

The Link header field defines several target attributes specific to this serialisation, and also allows extension target attributes. Target attributes are serialised in the Link header field as parameters (see [\[RFC7231\]](#), [Section 3.1.1.1](#) for the definition of their syntax).

3.4.1. Serialisation-Defined Attributes

The "hreflang", "media", "title", "title*", and "type" link-params can be translated to serialisation-defined target attributes for the link.

The "hreflang" attribute, when present, is a hint indicating what the language of the result of dereferencing the link should be. Note that this is only a hint; for example, it does not override the Content-Language header field of a HTTP response obtained by actually following the link. Multiple "hreflang" attributes on a single link-value indicate that multiple languages are available from the indicated resource.

The ABNF for the "hreflang" parameter's value is:

Language-Tag

The "media" attribute, when present, is used to indicate intended destination medium or media for style information (see [\[W3C.REC-html5-20141028\]](#), Section 4.2.4). Its value MUST be quoted if it contains a semicolon (";") or comma (","). There MUST NOT be more than one "media" attribute in a link-value; occurrences after the first MUST be ignored by parsers.

The ABNF for the "media" parameter's value is:

media-query-list

The "title" attribute, when present, is used to label the destination of a link such that it can be used as a human-readable identifier (e.g., a menu entry) in the language indicated by the Content-Language header field (if present). The "title" attribute MUST NOT appear more than once in a given link; occurrences after the first MUST be ignored by parsers.

The "title*" link-param can be used to encode this attribute in a different character set, and/or contain language information as per [\[I-D.ietf-httpbis-rfc5987bis\]](#). The "title*" link-param MUST NOT appear more than once in a given link-value; occurrences after the

Nottingham

Expires February 12, 2018

[Page 11]

first MUST be ignored by parsers. If the attribute does not contain language information, its language is indicated by the Content-Language header field (when present).

If both the "title" and "title*" link-param appear in a link, applications SHOULD use the "title*" link-param's value for the "title" attribute.

The "type" attribute, when present, is a hint indicating what the media type of the result of dereferencing the link should be. Note that this is only a hint; for example, it does not override the Content-Type header field of a HTTP response obtained by actually following the link. The "type" attribute MUST NOT appear more than once in a given link-value; occurrences after the first MUST be ignored by parsers.

The ABNF for the "type" parameter's value is:

type-name "/" subtype-name ; see [{{RFC6838}}](#), [Section 4.2](#)

[3.4.2.](#) Extension Attributes

Other link-params are link-extensions, and are to be considered as target attributes.

Such target attributes MAY be defined to use the encoding in [\[I-D.ietf-httpbis-rfc5987bis\]](#) (e.g., "example" and "example*"). When both forms are present, they SHOULD be considered to be the same target attribute; applications SHOULD use the value of the name ending in "*" (after [\[I-D.ietf-httpbis-rfc5987bis\]](#) decoding), but MAY fall back to the other value if there is an error in decoding it, or if they do not support decoding.

[3.5.](#) Link Header Field Examples

For example:

```
Link: <http://example.com/TheBook/chapter2>; rel="previous";  
      title="previous chapter"
```

indicates that "chapter2" is previous to this resource in a logical navigation path.

Similarly,

```
Link: </>; rel="http://example.net/foo"
```

Nottingham

Expires February 12, 2018

[Page 12]

indicates that the root resource ("/") is related to this resource with the extension relation type "http://example.net/foo".

This link:

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

indicates that the linked copyright terms only apply to the portion of the document indicated by the (media type-specific) fragment identifier "foo".

The example below shows an instance of the Link header field encoding multiple links, and also the use of [RFC 5987](#) encoding to encode both non-ASCII characters and language information.

```
Link: </TheBook/chapter2>;  
      rel="previous"; title*=UTF-8'de'letztes%20Kapitel,  
      </TheBook/chapter4>;  
      rel="next"; title*=UTF-8'de'n%c3%a4chstes%20Kapitel
```

Here, both links have titles encoded in UTF-8, use the German language ("de"), and the second link contains the Unicode code point U+00E4 ("LATIN SMALL LETTER A WITH DIAERESIS").

Note that link-values can convey multiple links between the same link target and link context; for example:

```
Link: <http://example.org/>;  
      rel="start http://example.net/relation/other"
```

Here, the link to "http://example.org/" has the registered relation type "start" and the extension relation type "http://example.net/relation/other".

Finally, this header field:

```
Link: <https://example.org/>; rel="start",  
      <https://example.org/index>; rel="index"
```

is equivalent to these:

```
Link: <https://example.org/>; rel="start"  
Link: <https://example.org/index>; rel="index"
```


4. IANA Considerations

4.1. Link HTTP Header Field Registration

This specification updates the Message Header registry entry for "Link" in HTTP [[RFC3864](#)] to refer to this document.

Header field: Link

Applicable protocol: http

Status: standard

Author/change controller:

IETF (iesg@ietf.org)

Internet Engineering Task Force

Specification document(s):

[this document]

4.2. Link Relation Type Registry

This specification updates the registration procedures for the Link Relation Type registry; see [Section 2.1.1.1](#).

IANA will direct any incoming requests regarding the registry to this document and, if defined, the processes established by the expert(s); typically, this will mean referring them to the registry Web page.

Note that the expert(s) are allowed (as per [Section 2.1.1.1](#)) to define additional fields to be collected in the registry.

4.3. Link Relation Application Data Registry

This specification removes the Link Relation Application Data Registry, as it has not been used, and future use is not anticipated. IANA is instructed to remove it.

5. Security Considerations

The content of the Link header field is not secure, private or integrity-guaranteed. Use of Transport Layer Security (TLS) with HTTP ([\[RFC2818\]](#)) is currently the only end-to-end way to provide these properties.

Link applications ought to consider the attack vectors opened by automatically following, trusting, or otherwise using links gathered from HTTP header fields.

For example, Link header fields that use the "anchor" parameter to associate a link's context with another resource cannot be trusted since they are effectively assertions by a third party that could be

Nottingham

Expires February 12, 2018

[Page 14]

incorrect or malicious. Applications can mitigate this risk by specifying that such links should be discarded unless some relationship between the resources is established (e.g., they share the same authority).

Dereferencing links has a number of risks, depending on the application in use. For example, the Referer header [[RFC7231](#)] can expose information about the application's state (including private information) in its value. Likewise, cookies [[RFC6265](#)] are another mechanism that, if used, can become an attack vector. Applications can mitigate these risks by carefully specifying how such mechanisms should operate.

The Link header field makes extensive use of IRIs and URIs. See [[RFC3987](#)] [Section 8](#) for security considerations relating to IRIs. See [[RFC3986](#)] [Section 7](#) for security considerations relating to URIs. See [[RFC7230](#)] [Section 9](#) for security considerations relating to HTTP header fields.

6. Internationalisation Considerations

Link targets may need to be converted to URIs in order to express them in serialisations that do not support IRIs. This includes the Link HTTP header field.

Similarly, the anchor parameter of the Link header field does not support IRIs, and therefore IRIs must be converted to URIs before inclusion there.

Relation types are defined as URIs, not IRIs, to aid in their comparison. It is not expected that they will be displayed to end users.

Note that registered Relation Names are required to be lower-case ASCII letters.

7. References

7.1. Normative References

[I-D.ietf-httpbis-rfc5987bis]
Reschke, J., "Indicating Character Encoding and Language for HTTP Header Field Parameters", [draft-ietf-httpbis-rfc5987bis-05](#) (work in progress), February 2017.

Nottingham

Expires February 12, 2018

[Page 15]

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", [BCP 90](#), [RFC 3864](#), DOI 10.17487/RFC3864, September 2004, <<http://www.rfc-editor.org/info/rfc3864>>.
- [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, <<http://www.rfc-editor.org/info/rfc3986>>.
- [RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", [RFC 3987](#), DOI 10.17487/RFC3987, January 2005, <<http://www.rfc-editor.org/info/rfc3987>>.
- [RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", [BCP 47](#), [RFC 5646](#), DOI 10.17487/RFC5646, September 2009, <<http://www.rfc-editor.org/info/rfc5646>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", [BCP 13](#), [RFC 6838](#), DOI 10.17487/RFC6838, January 2013, <<http://www.rfc-editor.org/info/rfc6838>>.
- [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, <<http://www.rfc-editor.org/info/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, <<http://www.rfc-editor.org/info/rfc7231>>.
- [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, <<http://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<http://www.rfc-editor.org/info/rfc8174>>.

Nottingham

Expires February 12, 2018

[Page 16]

[W3C.REC-css3-mediaqueries-20120619]

Rivoal, F., "Media Queries", World Wide Web Consortium Recommendation REC-css3-mediaqueries-20120619, June 2012, <<http://www.w3.org/TR/2012/REC-css3-mediaqueries-20120619>>.

7.2. Informative References

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", [RFC 2046](#), DOI 10.17487/RFC2046, November 1996, <<http://www.rfc-editor.org/info/rfc2046>>.

[RFC2818] Rescorla, E., "HTTP Over TLS", [RFC 2818](#), DOI 10.17487/RFC2818, May 2000, <<http://www.rfc-editor.org/info/rfc2818>>.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", [RFC 4287](#), DOI 10.17487/RFC4287, December 2005, <<http://www.rfc-editor.org/info/rfc4287>>.

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

[W3C.REC-html5-20141028]

Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Navara, E., O'Connor, T., and S. Pfeiffer, "HTML5", World Wide Web Consortium Recommendation REC-html5-20141028, October 2014, <<http://www.w3.org/TR/2014/REC-html5-20141028>>.

[Appendix A](#). Notes on Other Link Serialisations

Header fields ([Section 3](#)) are only one serialisation of links; other specifications have defined alternative serialisations.

[A.1](#). Link Serialisation in HTML

HTML motivated the original syntax of the Link header field, and many of the design decisions in this document are driven by a desire to stay compatible with it.

In HTML, the link element can be mapped to links as specified here by using the "href" attribute for the target URI, and "rel" to convey the relation type, as in the Link header field. The context of the link is the URI associated with the entire HTML document. HTML also defines several attributes on links that can be seen as target attributes, including "media", "hreflang", "type" and "sizes".

[Section 4.8](#) of HTML5 ([\[W3C.REC-html5-20141028\]](#)) defines modern HTML links. That document links to the Microformats Wiki as a registry; over time, the IANA registry ought to mirror its contents, and ideally eventually replace it (although that depends on the HTML community).

Surveys of existing HTML content have shown that unregistered link relation types that are not URIs are (perhaps inevitably) common. Consuming HTML implementations ought not consider such unregistered short links to be errors, but rather relation types with a local scope (i.e., their meaning is specific and perhaps private to that document).

Finally, the HTML specification gives a special meaning when the "alternate" relation types coincides with other relation types in the same link. Such links ought to be serialised in the Link header field using a single list of relation-types (e.g., rel="alternate stylesheet") to preserve this relationship.

[A.2](#). Link Serialisation in Atom

Atom [\[RFC4287\]](#) is a link serialisation that conveys links in the atom:link element, with the "href" attribute indicating the link target and the "rel" attribute containing the relation type. The context of the link is either a feed locator or an entry ID, depending on where it appears; generally, feed-level links are obvious candidates for transmission as a Link header field.

When serialising an atom:link into a Link header field, it is necessary to convert link targets (if used) to URIs.

Nottingham

Expires February 12, 2018

[Page 18]

Atom defines extension relation types in terms of IRIs. This specification re-defines them as URIs, to simplify and reduce errors in their comparison.

Atom allows registered link relation types to be serialised as absolute URIs using a prefix, "http://www.iana.org/assignments/relation/". This prefix is specific to the Atom serialisation.

Furthermore, link relation types are always compared in a case-sensitive fashion; therefore, registered link relation types SHOULD be converted to their registered form (usually, lowercase) when serialised in an Atom document.

Note also that while the Link header field allows multiple relations to be serialised in a single link, atom:link does not. In this case, a single link-value may map to several atom:link elements.

As with HTML, atom:link defines some attributes that are not explicitly mirrored in the Link header field syntax, but they can also be used as link-extensions to maintain fidelity.

[Appendix B](#). Algorithms for Parsing Link Header Fields

This appendix outlines a set of non-normative algorithms: for parsing the Link header(s) out of a header set, parsing a link header field value, and algorithms for parsing generic parts of the field value.

These algorithms are more permissive than the ABNF defining the syntax might suggest; the error handling embodied in them is a reasonable approach, but not one that is required. As such they are advisory only, and in cases where there is disagreement, the correct behaviour is defined by the body of this specification.

[B.1](#). Parsing a Header Set for Links

This algorithm can be used to parse the Link header fields that a HTTP header set contains. Given a "header_set" of (string "field_name", string "field_value") pairs, assuming ASCII encoding, it returns a list of link objects.

1. Let "field_values" be a list containing the members of "header_set" whose "field_name" is a case-insensitive match for "link".
2. Let "links" be an empty list.
3. For each "field_value" in "field_values":

Nottingham

Expires February 12, 2018

[Page 19]

1. Let "value_links" be the result of `_Parsing A Link Field Value_` (Appendix B.2) from "field_value".
2. Append each member of "value_links" to "links".
4. Return "links".

B.2. Parsing a Link Field Value

This algorithm parses zero or more comma-separated link-values from a Link header field. Given a string "field_value", assuming ASCII encoding, it returns a list of link objects.

1. Let "links" be an empty list.
2. While "field_value" has content:
 1. Consume any leading OWS.
 2. If the first character is not "<", return "links".
 3. Discard the first character ("<").
 4. Consume up to but not including the first ">" character or end of "field_value" and let the result be "target_string".
 5. If the next character is not ">", return "links".
 6. Discard the leading ">" character.
 7. Let "link_parameters", be the result of `_Parsing Parameters_` (Appendix B.3) from "field_value" (consuming zero or more characters of it).
 8. Let "target" be the result of relatively resolving (as per [\[RFC3986, Section 5.2\]](#)) "target_string". Note that any base URI carried in the payload body is NOT used.
 9. Let "relations_string" be the second item of the first tuple of "link_parameters" whose first item matches the string "rel", or the empty string ("") if it is not present.
 10. Split "relations_string" on RWS (removing it in the process) into a list of strings "relation_types".
 11. Let "context_string" be the second item of the first tuple of "link_parameters" whose first item matches the string "anchor". If it is not present, "context_string" is the URL

of the representation carrying the Link header [\[RFC7231\]](#), [Section 3.1.4.1](#), serialised as a URI. Where the URL is anonymous, "context_string" is null.

12. Let "context" be the result of relatively resolving (as per [\[RFC3986\]](#), [Section 5.2](#)) "context_string", unless "context_string" is null in which case "context" is null. Note that any base URI carried in the payload body is NOT used.
13. Let "target_attributes" be an empty list.
14. For each tuple ("param_name", "param_value") of "link_parameters":
 1. If "param_name" matches "rel" or "anchor", skip this tuple.
 2. If "param_name" matches "media", "title", "title*" or "type" and "target_attributes" already contains a tuple whose first element matches the value of "param_name", skip this tuple.
 3. Append ("param_name", "param_value") to "target_attributes".
15. Let "star_param_names" be the set of "param_name"s in the ("param_name", "param_value") tuples of "link_parameters" where the last character of "param_name" is an asterisk (*).
16. For each "star_param_name" in "star_param_names":
 1. Let "base_param_name" be "star_param_name" with the last character removed.
 2. If the implementation does not choose to support an internationalised form of a parameter named "base_param_name" for any reason (including, but not limited to, it being prohibited by the parameter's specification), remove all tuples from "link_parameters" whose first member is "star_param_name" and skip to the next "star_param_name".
 3. Remove all tuples from "link_parameters" whose first member is "base_param_name".

Nottingham

Expires February 12, 2018

[Page 21]

4. Change the first member of all tuples in "link_parameters" whose first member is "star_param_name" to "base_param_name".
17. For each "relation_type" in "relation_types":
 1. Case-normalise "relation_type" to lowercase.
 2. Append a link object to "links" with the target "target", relation type of "relation_type", context of "context", and target attributes "target_attributes".
3. Return "links".

B.3. Parsing Parameters

This algorithm parses the parameters from a header field value. Given an ASCII string "input", it returns a list of (string "parameter_name", string "parameter_value") tuples that it contains. "input" is modified to remove the parsed parameters.

1. Let "parameters" be an empty list.
2. While "input" has content:
 1. Consume any leading OWS.
 2. If the first character is not ";", return "parameters".
 3. Discard the leading ";" character.
 4. Consume any leading OWS.
 5. Consume up to but not including the first BWS, "=", ";", ",", " character or end of "input" and let the result be "parameter_name".
 6. Consume any leading BWS.
 7. If the next character is "=":
 1. Discard the leading "=" character.
 2. Consume any leading BWS.
 3. If the next character is DQUOTE, let "parameter_value" be the result of `_Parsing a Quoted String`

(Appendix B.4) from "input" (consuming zero or more characters of it).

4. Else, consume the contents up to but not including the first ";", ",", " character or end of "input" and let the results be "parameter_value".
5. If the last character of "parameter_name" is an asterisk ("*"), decode "parameter_value" according to [\[I-D.ietf-httpbis-rfc5987bis\]](#). Continue processing "input" if an unrecoverable error is encountered.
8. Else:
 1. Let "parameter_value" be an empty string.
9. Case-normalise "parameter_name" to lowercase.
10. Append ("parameter_name", "parameter_value") to "parameters".
11. Consume any leading OWS.
12. If the next character is ",", or the end of "input", stop processing "input" and return "parameters".

[B.4.](#) Parsing a Quoted String

This algorithm parses a quoted string, as per [\[RFC7230\]](#), [Section 3.2.6](#). Given an ASCII string "input", it returns an unquoted string. "input" is modified to remove the parsed string.

1. Let "output" be an empty string.
2. If the first character of "input" is not DQUOTE, return "output".
3. Discard the first character.
4. While "input" has content:
 1. If the first character is a backslash ("\"):
 1. Discard the first character.
 2. If there is no more "input", return "output".
 3. Else, consume the first character and append it to "output".

2. Else, if the first character is DQUOTE, discard it and return "output".
3. Else, consume the first character and append it to "output".
5. Return "output".

Appendix C. Changes from [RFC5988](#)

This specification has the following differences from its predecessor, [RFC5988](#):

- o The initial relation type registrations were removed, since they've already been registered by 5988.
- o The introduction has been shortened.
- o The Link Relation Application Data Registry has been removed.
- o Incorporated errata.
- o Updated references.
- o Link cardinality was clarified.
- o Terminology was changed from "target IRI" and "context IRI" to "link target" and "link context" respectively.
- o Made assigning a URI to registered relation types serialisation-specific.
- o Removed misleading statement that the link header field is semantically equivalent to HTML and Atom links.
- o More carefully defined and used "link serialisations" and "link applications."
- o Clarified the cardinality of target attributes (generically and for "type").
- o Corrected the default link context for the Link header field, to be dependent upon the identity of the representation (as per [RFC7231](#)).
- o Defined a suggested parsing algorithm for the Link header.
- o The value space of target attributes and their definition has been specified.

- o The ABNF has been updated to be compatible with [[RFC7230](#)]. In particular, whitespace is now explicit.
- o Some parameters on the HTTP header field can now appear as a token.
- o Parameters on the HTTP header can now be value-less.
- o Handling of quoted strings is now defined by [[RFC7230](#)].
- o The "type" header field parameter now needs to be quoted (as "token" does not allow "/").

Author's Address

Mark Nottingham

EMail: mnot@mnot.net

URI: <https://www.mnot.net/>

