Web Linking

draft-nottingham-rfc5988bis-04

Abstract

This specification defines a way to indicate 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

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The most recent (often, unpublished) draft is at https://mnot.github.io/I-D/rfc5988bis/.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 August 7, 2017.
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 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
2. Notational Conventions ..................................... 3
3. Links ....................................................... 4
4. Link Relation Types ........................................ 5
   4.1. Registered Relation Types ............................ 5
       4.1.1. Registering Link Relation Types .................. 6
       4.1.2. Registration Request Processing .................. 7
   4.2. Extension Relation Types .............................. 7
5. Target Attributes ........................................... 8
6. Link Serialisation in HTTP Headers ......................... 8
   6.1. Link Target ........................................... 9
   6.2. Link Context .......................................... 9
   6.3. Relation Type ......................................... 9
   6.4. Target Attributes .................................... 10
       6.4.1. Serialisation-Defined Attributes ................ 10
       6.4.2. Extension Attributes ............................ 11
   6.5. Examples .............................................. 12
7. IANA Considerations ....................................... 13
   7.1. Link HTTP Header Field Registration ................ 13
1. Introduction

This specification defines a way to indicate 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; this specification generalises this into a framework that encompasses linking in these formats and (potentially) elsewhere.

Furthermore, this specification formalises an HTTP header field for conveying such links, having been originally defined in Section 19.6.2.4 of [RFC2068], but removed from [RFC2616].

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119], as scoped to those conformance targets.

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.

Additionally, the following rules are included from [RFC3986]: URI and URI-Reference; from [RFC6838]: type-name and subtype-name; from [W3C.CR-css3-mediaqueries-20090915]: media_query_list; and from [RFC5646]: Language-Tag..
3. Links

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

- A _link context_,
- a _link relation type_ (Section 4),
- a _link target_, and
- optionally, _target attributes_ (Section 5).

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}".

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 places that do not support IRIs (such as the Link header field defined in Section 6).

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 [RFC4287] and HTML [W3C.REC-html5-20141028] both defined serialisations of links into their respective formats, and Section 6 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 consumed by _link applications_. Generally, an application will define the link relation types it uses, along with the serialisations that they might occur within. For example, the application "Web browsing" looks for the "stylesheet" link relation
type in the HTML link serialisation, whereas the application
"AtomPub" uses the "edit" and "edit-media" link relations.

4. 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 resource identified by the link target is a
statement of the copyright terms applying to the current link
context.

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 identified
resource is part of a defined protocol (in this case, a service
description).

Relation types are not to be confused with media types [RFC6838];
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.

4.1. 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 4.1.1.

Registered relation type names MUST conform to the reg-rel-type rule
(see Section 6.3), 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 must 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 processors. Applications that do use such URIs internally MUST NOT use them in link serialisations that do not explicitly accommodate them.

4.1.1. Registering Link Relation Types

Any party can request registration of a link relation type.

Registration requests can be sent to the "link-relations@ietf.org" mailing list. The Expert(s) MAY establish alternate means of requesting registrations, which SHOULD be linked to from the registry page.

Registration requests consist of at least the following information:

- *Relation Name*: The name of the relation type
- *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.
- *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) MAY also be included, but is not required.

The Expert(s) MAY define additional fields to be collected in the registry.

General requirements for registered relation types are described in Section 4.1.

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.
4.1.2. Registration Request Processing

Relation types are registered on the advice of a Designated Expert (appointed by the IESG or their delegate), with a Specification Required (using terminology from [RFC5226]).

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).

The Expert(s) MUST clearly identify any issues which cause a registration to be refused. Advice about the syntax or 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.

4.2. 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 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.
5. 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. Serialisations SHOULD coordinate their target attributes to avoid conflicts in semantics or syntax.

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:

- The serialisation of their values into Unicode or a subset thereof, to maximise their chances of portability across link serialisations.
- 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 6.4._

6. 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 given below:

```
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. 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 defines the link-params "rel", "anchor", "rev", "hreflang", "media", "title", "title*", and "type"; see Section 6.2, Section 6.3 and Section 6.4.

6.1.  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 from the message's content is not applied.

6.2.  Link Context

By default, the context of a link conveyed in the Link header field is identity 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:

    URI-Reference

Consuming implementations 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; consuming implementations 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 instance, this is the case on a 404 response to a GET request.

6.3.  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 NOT appear more than once in a given link-value; occurrences after the first MUST be ignored by parsers.
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
```

Note that extension relation types are REQUIRED to be absolute URIs in Link headers, and MUST be quoted if they contain a semicolon (";") or comma (",")) (as these characters are used as delimiters in the header field itself).

### 6.4. 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).

#### 6.4.1. 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
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,
processors 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

6.4.2. 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; processors 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.

6.5. 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"

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

In addition to the actions below, IANA should terminate the Link Relation Application Data Registry, as it has not been used, and future use is not anticipated.

7.1.  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):
  [RFC&rfc.number;]

7.2.  Link Relation Type Registry

This specification updates the registration procedures for the Link Relation Type registry; see Section 4.1.1. The Expert(s) and IANA will interact as outlined below.

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.

The Expert(s) will provide registry data to IANA in an agreed form (e.g. a specific XML format). IANA will publish:

o  The raw registry data

o  The registry data, transformed into HTML

o  The registry data in any alternative formats provided by the Expert(s)

Each published document will be at a URL agreed to by IANA and the Expert(s), and IANA will set HTTP response headers on them as (reasonably) requested by the Expert(s).

Additionally, the HTML generated by IANA will:

o  Take directions from the Expert(s) as to the content of the HTML page's introductory text
Include a stable HTML fragment identifier for each registered link relation

All registry data documents MUST include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions (<http://trustee.ietf.org/license-info>).

8. Security Considerations

The content of the Link header field is not secure, private or integrity-guaranteed, and due caution should be exercised when using it. Use of Transport Layer Security (TLS) with HTTP ([RFC2818] and [RFC2817]) is currently the only end-to-end way to provide such protection.

Link applications ought to consider the attack vectors opened by automatically following, trusting, or otherwise using links gathered from HTTP headers. In particular, Link header fields that use the "anchor" parameter to associate a link's context with another resource should be treated with due caution.

The Link header field makes extensive use of IRIs and URIs. See [RFC3987] for security considerations relating to IRIs. See [RFC3986] for security considerations relating to URIs. See [RFC7230] for security considerations relating to HTTP headers.

9. 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.

10. References
10.1. Normative References

[I-D.ietf-httpbis-rfc5987bis]


10.2. Informative References


Appendix A. Notes on Other Link Serialisations

Header fields (Section 6) 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".

HTML5 ([W3C.REC-html5-20141028]) Section 4.8 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.
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. Algorithm for Parsing Link Headers

Given a HTTP header field-value "field_value" as a string assuming ASCII encoding, the following algorithm can be used to parse it into the model described by this specification:

1. Let "links" be an empty list.

2. Create "link_strings" by splitting "field_value" on "," characters, excepting "," characters within quoted strings as per [RFC7230], Section 3.2.6, or which form part of link's URI-Reference (i.e. between "<" and ">" characters where the "<" is immediately preceded by OWS and either a "," character or the beginning of the "field_value" string).

3. For each "link_string" in "link_strings":
   1. Let "target_string" be the string between the first "<" and first ">" characters in "link_string". If they do not appear, or do not appear in that order, fail parsing.
   2. Let "rest" be the remaining characters (if any) after the first ">" character in "link_string".
   3. Split "rest" into an array of strings "parameter_strings", on the ";" character, excepting ";" characters within quoted strings as per [RFC7230], Section 3.2.6.
4. Let "link_parameters" be an empty array.

5. For each item "parameter" in "parameter_strings":
   1. Remove OWS from the beginning and end of "parameter".
   2. Skip this item if "parameter" matches the empty string ("")
   3. Split "parameter" into "param_name" and "param_value" on the first "=" character. If "parameter" does not contain "=" let "param_name" be "parameter" and "param_value" be null.
   4. Remove OWS from the end of "param_name" and the beginning of "param_value".
   5. Case-normalise "param_name" to lowercase.
   6. If the first and last characters of "param_value" are both DQUOTE:
      1. Remove the first and last characters of "param_value".
      2. Replace quoted-pairs within "param_value" with the octet following the backslash, as per [RFC7230], Section 3.2.6.
   7. If the last character of "param_name" is an asterisk ("*"), decode "param_value" according to [I-D.ietf-httpbis-rfc5987bis]. Skip this item if an unrecoverable error is encountered.
   8. Append the tuple ("param_name", "param_value") to "link_parameters".

6. 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.

7. 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.

8. Split "relations_string" into an array of strings "relation_types", on RWS (removing all whitespace in the process).
9. 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 identity of the representation carrying the Link header [RFC7231], Section 3.1.4.1, serialised as a URI. Where the identity is "anonymous" "context_string" is null.

10. 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.

11. Let "target_attributes" be an empty array.

12. 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".

13. 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 ("*").

14. 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".

4. Change the first member of all tuples in "link_parameters" whose first member is "star_param_name" to "base_param_name".

15. 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".

4. Return "links".

Appendix C. Changes from RFC5988

This specification has the following differences from its predecessor, RFC5988:

- The initial relation type registrations were removed, since they've already been registered by 5988.
- The introduction has been shortened.
- The Link Relation Application Data Registry has been removed.
- Incorporated errata.
- Updated references.
- Link cardinality was clarified.
- Terminology was changed from "target IRI" and "context IRI" to "link target" and "link context" respectively.
- Made assigning a URI to registered relation types serialisation-specific.
- Removed misleading statement that the link header field is semantically equivalent to HTML and Atom links.
- More carefully defined how the Experts and IANA should interact.
- More carefully defined and used "link serialisations" and "link applications."
Clarified the cardinality of target attributes (generically and for "type").

Corrected the default link context for the Link header field, to be dependent upon the identity of the representation (as per RFC7231).

Defined a suggested parsing algorithm for the Link header.

The value space of target attributes and their definition has been specified.

The ABNF has been updated to be compatible with [RFC7230]. In particular, whitespace is now explicit.

Some parameters on the HTTP header field can now appear as a token.

Handling of quoted strings is now defined by [RFC7230].

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/