Network Working Group   E. Wilde
Internet-Draft   Axway
Intended status: Informational   H. Van de Sompel
Expires: December 6, 2021   Data Archiving and Networked Services
                                                          June 4, 2021

Linkset: Media Types and a Link Relation Type for Link Sets
draft-ietf-httpapi-linkset-02

Abstract

This specification defines two document formats and respective media types for representing sets of links as stand-alone resources. One format is JSON-based, the other aligned with the format for representing links in the HTTP "Link" header field. This specification also introduces a link relation type to support discovery of sets of links.

Note to Readers

Please discuss this draft on the "Building Blocks for HTTP APIs" mailing list (<https://www.ietf.org/mailman/listinfo/httpapi>).

Online access to all versions and files is available on GitHub (<https://github.com/ietf-wg-httpapi/linkset>).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 6, 2021.

Table of Contents

1. Introduction

Resources on the Web often use typed Web Links [RFC8288], either embedded in resource representations, for example using the <link> element for HTML documents, or conveyed in the HTTP "Link" header for documents of any media type. In some cases, however, providing links in this manner is impractical or impossible and delivering a set of links as a stand-alone document is preferable.

Therefore, this specification defines two document formats and associated media types to represent sets of links. It also defines the "linkset" relation type that supports discovery of any resource that conveys a set of links as a stand-alone document.

2. Terminology

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 specification uses the terms "link context" and "link target" as defined in [RFC8288]. These terms respectively correspond with "Context IRI" and "Target IRI" as used in [RFC5988]. Although defined as IRIs, in common scenarios they are also URIs.

In the examples provided in this document, links in the HTTP "Link" header are shown on separate lines in order to improve readability. Note, however, that as per Section 3.2 of [RFC7230], line breaks are not allowed in values for HTTP headers; only whitespaces and tabs are supported as seperators.

3. Scenarios

The following sections outline scenarios in which providing links by means of a standalone document instead of in an HTTP "Link" header field or as links embedded in the resource representation is advantageous or necessary.

For all scenarios, links could be provided by means of a stand-alone document that is formatted according to the JSON-based serialization, the serialization aligned with the HTTP "Link" header format, or both. The former serialization is motivated by the widespread use of JSON and related tools, which suggests that handling sets of links expressed as JSON documents should be attractive to developers. The latter serialization is provided for compatibility with the existing serialization used in the HTTP "Link" header and to allow reuse of tools created to handle it.

It is important to keep in mind that when providing links by means of a standalone representation, other links can still be provided using other approaches, i.e. it is possible combine various mechanisms to convey links.

3.1. Third-Party Links

In some cases it is useful that links pertaining to a resource are provided by a server other than the one that hosts the resource. For example, this allows:

- Providing links in which the resource is involved not just as link context but also as link target.
- Providing links pertaining to the resource that the server hosting that resource is not aware of.
- External management of links pertaining to the resource in a special-purpose link management service.

In such cases, links pertaining to a resource can be provided by another, specific resource. That specific resource may be managed by the same or by another custodian as the resource to which the links pertain. For clients intent on consuming links provided in that manner, it would be beneficial if the following conditions were met:

- Links are provided in a document that uses a well-defined media type.
- The resource to which the provided links pertain is able to link to the resource that provides these links using a well-known link relation type.

These requirements are addressed in this specification through the definition of two media types and a link relation type, respectively.

3.2. Challenges Writing to HTTP Link Header Field

In some cases, it is not straightforward to write links to the HTTP "Link" header field from an application. This can, for example, be the case because not all required link information is available to the application or because the application does not have the capability to directly write HTTP headers. In such cases, providing links by means of a standalone document can be a solution. Making the resource that provides these links discoverable can be achieved by means of a typed link.

3.3. Large Number of Links

When conveying links in an HTTP "Link" header field, it is possible for the size of the HTTP response header to become unpredictable. This can be the case when links are determined dynamically dependent on a range of contextual factors. It is possible to statically configure a web server to correctly handle large HTTP response headers by specifying an upper bound for their size. But when the number of links is unpredictable, estimating a reliable upper bound is challenging.

HTTP [RFC7231] defines error codes related to excess communication by the user agent ("413 Request Entity Too Large" and "414 Request-URI Too Long"), but no specific error codes are defined to indicate that response header content exceeds the upper bound that can be handled by the server, and thus it has been truncated. As a result, applications take counter measures aimed at controlling the size of the HTTP "Link" header field, for example by limiting the links they provide to those with select relation types, thereby limiting the value of the HTTP "Link" header field to clients. Providing links by means of a standalone document overcomes challenges related to the unpredictable nature of the size of HTTP "Link" header fields.

4. Document Formats for Sets of Links

This section specifies two document formats to convey a set of links. Both are based on the abstract model specified in Section 2 of Web Linking [RFC8288] that defines a link as consisting of a "link context", a "link relation type", a "link target", and optional "target attributes":

- The format defined in Section 4.1 is identical to the payload of the HTTP "Link" header field as specified in Web Linking [RFC8288].

- The format defined in Section 4.2 is based on JSON [RFC8259].

Note that [RFC8288] deprecates the "rev" construct that was provided by [RFC5988] as a means to express links with a directionality that is the inverse of direct links that use the "rel" construct. In both serializations for link sets defined here, inverse links SHOULD be represented as direct links using the "rel" construct and by switching the position of the resources involved in the link.

4.1. HTTP Link Document Format: application/linkset

This document format is identical to the payload of the HTTP "Link" header field as defined in Section 3 of [RFC8288], more specifically by its ABNF production rule for "Link" and subsequent ones. Whereas the HTTP "Link" Header field depends on HTTP and hence on [RFC0822] for its encoding, the format specified here is encoded as UTF-8 [RFC3629].

The assigned media type for this format is "application/linkset".

In order to support use cases where "application/linkset" documents are re-used outside the context of an HTTP interaction, it is RECOMMENDED to make them self-contained by adhering to the following guidelines:

- For every link provided in the set of links, explicitly provide the link context using the "anchor" attribute.

- For link context ("anchor" attribute) and link target ("href" attribute), use absolute URIs (as defined in Section 4.3 of [RFC3986]).

If these recommendations are not followed, interpretation of links in "application/linkset" documents will depend on which URI is used as context.

It should be noted that the "application/linkset" format specified here is different than the "application/link-format" format specified in [RFC6690] in that the former fully matches the payload of the HTTP "Link" header as defined in Section 3 of [RFC8288], whereas the latter introduces constraints on that definition to meet requirements for Constrained RESTful Environments.

4.2. JSON Document Format: application/linkset+json

This document format uses JSON [RFC8259] as the syntax to represent a set of links. The set of links follows the abstract model defined by Web Linking [RFC8288].

The assigned media type for this format is "application/linkset+json".

In order to support use cases where "application/linkset+json" documents are re-used outside the context of an HTTP interaction, it is RECOMMENDED to make them self-contained by adhering to the following guidelines:

- For every link provided in the set of links, explicitly provide the link context using the "anchor" member.
- For link context ("anchor" member) and link target ("href" member), use absolute URIs (as defined in Section 4.3 of [RFC3986]).

If these recommendations are not followed, interpretation of "application/linkset+json" will depend on which URI is used as context URI.

The "application/linkset+json" serialization is designed such that it can directly be used as the content of a JSON-LD serialization by adding an appropriate context. Appendix B shows an example of a possible context that, when added to a JSON serialization, allows it to be interpreted as RDF.

4.2.1. Set of Links

In the JSON representation of a set of links:

- A set of links MUST be represented as a JSON object which MUST have "linkset" as its sole member.
- The "linkset" member is an array in which a distinct JSON object - the "link context object" (see Section 4.2.2) - MUST be used to represent links that have the same link context.
- If necessary, the "linkset" member MAY contain information in addition to link context objects, in which case that information MUST NOT change the semantics of the links provided by those link context objects.
- Even if there is only one link context object, it MUST be wrapped in an array. Members other than link context objects MUST NOT be included in this array.

4.2.2. Link Context Object

In the JSON representation one or more links that have the same link context are represented by a JSON object, the link context object. A link context object adheres to the following rules:

- Each link context object MAY have an "anchor" member with a value that represents the link context. If present, this value SHOULD be an absolute URI as defined in Section 4.3 of [RFC3986]. Cases where the anchor member is present, but no value is provided for it (i.e. the resource providing the set of links is the link context for each link in the link context object) MUST be handled by providing an "anchor" member with empty string ("anchor": "").

- For each distinct relation type that the link context has with link targets, a link context object MUST have an additional member. This member is an array in which a distinct JSON object - the "link target object" (see Section 4.2.3) - MUST be used for each link target for which the relationship with the link context (value of the encompassing anchor member) applies. The name of this member expresses the relation type of the link as follows:

-

  * For registered relation types [RFC8288], the name of this member is the registered name of the relation type.

  * For extension relation types [RFC8288], the name of this member is the URI that uniquely represents the relation type.

- Even if there is only one link target object it MUST be wrapped in an array. Members other than link target objects MUST NOT be included in this array.

4.2.3. Link Target Object

In the JSON representation a link target is represented by a JSON object, the link target object. A link target object adheres to the following rules:

- Each link target object MUST have an "href" member with a value that represents the link target. This value SHOULD be an absolute URI as defined in Section 4.3 of [RFC3986]. Cases where the href member is present, but no value is provided for it (i.e. the resource providing the set of links is the target of the link in the link target object) MUST be handled by providing an "href" member with an empty string ("href": "").

- In many cases, a link target is further qualified by target attributes. Various types of attributes exist and they are conveyed as additional members of the link target object as detailed in Section 4.2.4.

The following example of a JSON-serialized set of links represents one link with its core components: link context, link relation type, and link target.

```
{
  "linkset":
    [
      { "anchor": "http://example.net/bar",
        "next": [
              {"href": "http://example.com/foo"}
        ]
      }
    ]
}
```

The following example of a JSON-serialized set of links represents two links that share link context and relation type but have different link targets.

```
{
  "linkset":
    [
      { "anchor": "http://example.net/bar",
        "item": [
              {"href": "http://example.com/foo1"},
              {"href": "http://example.com/foo2"}
        ]
      }
    ]
}
```

The following example shows a set of links that represents two links, each with a different link context, link target, and relation type. One relation type is registered, the other is an extension relation type.

```
{
  "linkset":
    [
      { "anchor": "http://example.net/bar",
        "next": [
              {"href": "http://example.com/foo1"}
        ]
      },
      { "anchor": "http://example.net/boo",
        "http://example.com/relations/baz" : [
              {"href": "http://example.com/foo2"}
        ]
      }
    ]
}
```

4.2.4. Link Target Attributes

A link may be further qualified by target attributes. Three types of attributes exist:

- Attributes defined by the serialization of Web Linking [RFC8288].
- Extension attributes defined and used by communities as allowed by [RFC8288].
- Internationalized versions of the "title" attribute defined by [RFC8288] and of extension attributes allowed by [RFC8288].

The handling of these different types of attributes is described in the sections below.

4.2.4.1. Target Attributes Defined by Web Linking

RFC 8288 defines the following target attributes that may be used to annotate links: "hreflang", "media", "title", "title*", and "type"; these target attributes follow different occurrence and value patterns. In the JSON representation, these attributes MUST be conveyed as additional members of the link target object as follows:

- "hreflang": The optional and repeatable "hreflang" target attribute MUST be represented by an array (even if there only is one value to be represented), and each value in that array MUST be a string - representing one value of the "hreflang" target attribute for a link - which follows the same model as in the [RFC8288] syntax.

- "media": The optional and not repeatable "media" target attribute MUST be represented by a "media" member in the link target object, and its value MUST be a string that follows the same model as in the [RFC8288] syntax.

- "type": The optional and not repeatable "type" target attribute MUST be represented by a "type" member in the link target object, and its value MUST be a string that follows the same model as in the [RFC8288] syntax.

- "title": The optional and not repeatable "title" target attribute MUST be represented by a "title" member in the link target object, and its value MUST be a string that follows the same model as in the [RFC8288] syntax.

- "title*": The optional and not repeatable "title*" target attribute is motivated by character encoding and language issues and follows the model defined in [RFC8187]. The details of the JSON representation that applies to title* are described in Section 4.2.4.2.

The following example illustrates how the repeatable "hreflang" and the not repeatable "type" target attributes are represented in a link target object.

```
{
  "linkset":
    [
      { "anchor": "http://example.net/bar",
        "next": [
              {"href":     "http://example.com/foo",
               "type":     "text/html",
               "hreflang": [ "en" , "de" ]
              }
        ]
      }
    ]
}
```

4.2.4.2. Internationalized Target Attributes

In addition to the target attributes described in Section 4.2.4.1, [RFC8288] also supports attributes that follow the content model of [RFC8187]. In [RFC8288], these target attributes are recognizable by the use of a trailing asterisk in the attribute name, such as "title*". The content model of [RFC8187] uses a string-based microsyntax that represents the character encoding, an optional

language tag, and the escaped attribute value encoded according to the specified character encoding.

The JSON serialization for these target attributes MUST be as follows:

- An internationalized target attribute is represented as a member of the link context object with the same name (including the *) of the attribute.

- The character encoding information as prescribed by [RFC8187] is not preserved; instead, the content of the internationalized attribute is represented in the character encoding used for the JSON set of links.

- The value of the internationalized target attribute is an array that contains one or more JSON objects. The name of one member of such JSON object is "value" and its value is the actual content (in its unescaped version) of the internationalized target attribute, i.e. the value of the attribute from which the encoding and language information are removed. The name of another, optional, member of such JSON object is "language" and its value is the language tag [RFC5646] for the language in which the attribute content is conveyed.

The following example illustrates how the "title*" target attribute defined by [RFC8288] is represented in a link target object.

```
{
  "linkset":
    [
      { "anchor": "http://example.net/bar",
        "next": [
              {"href":     "http://example.com/foo",
               "type":     "text/html",
               "hreflang": [ "en" , "de" ],
               "title":    "Next chapter",
               "title*":   [ { "value": "nachstes Kapitel" ,
                               "language" : "de" } ]
              }
        ]
      }
    ]
}
```

The above example assumes that the German title contains an umlaut character (in the native syntax it would be encoded as title*=UTF-8'de'n%c3%a4chstes%20Kapitel), which gets encoded in its unescaped

form in the JSON representation. This is not shown in the above example due to the limitations of RFC publication. Implementations MUST properly decode/encode internationalized target attributes that follow the model of [RFC8187] when transcoding between the "application/linkset" and the "application/linkset+json" formats.

4.2.4.3. Extension Target Attributes

Extension target attributes are attributes that are not defined by [RFC8288] (as listed in Section 4.2.4.1), but are nevertheless used to qualify links. They can be defined by communities in any way deemed necessary, and it is up to them to make sure their usage is understood by target applications. However, lacking standardization, there is no interoperable understanding of these extension attributes. One important consequence is that their cardinality is unknown to generic applications. Therefore, in the JSON serialization, all extension target attributes are treated as repeatable.

The JSON serialization for these target attributes MUST be as follows:

- An extension target attribute is represented as a member of the link context object with the same name of the attribute, including the * if applicable.

- The value of an extension attribute MUST be represented by an array, even if there only is one value to be represented.

- If the extension target attribute does not have a name with a trailing asterisk, then each value in that array MUST be a string that represents one value of the attribute.

- If the extension attribute has a name with a trailing asterisk (it follows the content model of [RFC8187]), then each value in that array MUST be a JSON object. The value of each such JSON object MUST be structured as described in Section 4.2.4.2.

The example shows a link target object with three extension target attributes. The value for each extension target attribute is an array. The two first are regular extension target attributes, with the first one ("foo") having only one value and the second one ("bar") having two. The last extension target attribute ("baz*") follows the naming rule of [RFC8187] and therefore is encoded according to the serialization described in Section 4.2.4.2.

```
{
  "linkset":
    [
      { "anchor": "http://example.net/bar",
        "next": [
              { "href": "http://example.com/foo",
                "type": "text/html",
                "foo":  [ "foovalue" ],
                "bar":  [ "barone", "bartwo" ],
                "baz*": [ { "value": "bazvalue" ,
                            "language" : "en" } ]
              }
        ]
      }
    ]
}
```

4.2.5. JSON Extensibility

The extensibility of the JSON document format for representing a set of links is restricted to the extensibility provided by [RFC8288]. The Web linking model provides for the use of extension target attributes as discussed in Section 4.2.4.3. Extensions based on the JSON syntax MUST NOT be used, and MUST be ignored when found in a JSON linkset document.

This limitation of the JSON format allows to unambiguously round trip between links provided in the HTTP "Link" header, sets of links serialized according to the "application/linkset" format, and sets of links serialized according to the "application/linkset+json" format.

5. The "profile" attribute for media types to Represent Sets of Links

As a means to convey specific constraints or conventions (as per [RFC6906]) that apply to a link set document, the "profile" attribute MAY be used in conjunction with the media types "application/linkset" and "application/linkset+json" detailed in Section 4.1 and Section 4.2, respectively. For example, the attribute could be used to indicate that a link set uses a specific, limited set of link relation types.

The value of the "profile" attribute MUST be a non-empty list of space-separated URIs, each of which identifies specific constraints or conventions that apply to the link set document. Profile URIs MAY be registered in the IANA Profile URI Registry in the manner specified by [RFC7284].

The presence of a "profile" attribute in conjunction with the "application/linkset" and "application/linkset+json" media types does not change the semantics of a link set. As such, clients with and without knowledge of profile URIs can use the same representation. The profile parameter MAY be used by clients to express their preferences, and, if a client does so, a server SHOULD return a document that honors the profiles it recognizes, and MUST ignore the profiles which it does not recognize.

6. The "linkset" Relation Type for Linking to a Set of Links

The target of a link with the "linkset" relation type provides a set of links, including links in which the resource that is the link context participates.

A link with the "linkset" relation type MAY be provided in the header and/or the body of a resource's representation. It may also be discovered by other means, such as through client-side information.

A resource MAY provide more than one link with a "linkset" relation type. Multiple such links can refer to the same set of links expressed using different media types, or to different sets of links, potentially provided by different third-party services.

A user agent that follows a "linkset" link MUST be aware that the set of links provided by the resource that is the target of the link can contain links in which the resource that is the context of the link does not participate; it MAY decide to ignore those links.

A user agent that follows a "linkset" link and obtains links for which anchors and targets are not expressed as absolute URIs MUST properly determine what the context is for these links; it SHOULD ignore links for which it is unable to unambiguously make that determination.

7. Examples

Section 7.1 and Section 7.2 show examples whereby a set of links is provided as "application/linkset" and "application/linkset+json" documents, respectively. Section 7.3 illustrates the use of the "linkset" link relation type to support discovery of sets of links.

7.1. Set of Links Provided as application/linkset

Figure 1 shows a client issuing an HTTP GET request against resource <https://example.org/links/resource1>.

```
GET /links/resource1 HTTP/1.1
Host: example.org
Connection: close
```

Figure 1: Client HTTP GET request

Figure 2 shows the response to the GET request of Figure 1. The response contains a Content-Type header specifying that the media type of the response is "application/linkset". A set of links, revealing authorship and versioning related to resource <https://example.org/resource1>, is provided in the response body. The HTTP "Link" header indicates the availability of an alternate representation of the set of links using media type "application/linkset+json".

```
HTTP/1.1 200 OK
Date: Mon, 12 Aug 2019 10:35:51 GMT
Server: Apache-Coyote/1.1
Content-Length: 1023
Content-Type: application/linkset; charset=UTF-8
Link: <https://example.org/links/resource1> ; rel="alternate"
; type="application/linkset+json"
Connection: close
<https://authors.example.net/johndoe>
   ; rel="author"
   ; type="application/rdf+xml"
   ; anchor="https://example.org/resource1",
<https://example.org/resource1?version=3>
   ; rel="latest-version"
   ; type="text/html"
   ; anchor="https://example.org/resource1",
<https://example.org/resource1?version=2>
   ; rel="predecessor-version"
   ; type="text/html"
   ; anchor="https://example.org/resource1?version=3",
<https://example.org/resource1?version=1>
   ; rel="predecessor-version"
   ; type="text/html"
   ; anchor="https://example.org/resource1?version=2",
<https://example.org/resource1?version=1>
   ; rel="memento"
   ; type="text/html"
   ; datetime="Thu, 13 Jun 2019 09:34:33 GMT"
   ; anchor="https://example.org/resource1",
<https://example.org/resource1?version=2>
   ; rel="memento"
   ; type="text/html"
   ; datetime="Sun, 21 Jul 2019 12:22:04 GMT"
   ; anchor="https://example.org/resource1",
<https://authors.example.net/alice>
   ; rel="author"
   ; anchor="https://example.org/resource1#comment=1"
```

Figure 2: Response to HTTP GET includes a set of links

7.2. Set of Links Provided as application/linkset+json

Figure 3 shows the client issuing an HTTP GET request against <https://example.org/links/resource1>. In the request, the client uses an "Accept" header to indicate it prefers a response in the "application/linkset+json" format.

```
GET links/resource1 HTTP/1.1
Host: example.org
Accept: application/linkset+json
Connection: close
```

Figure 3: Client HTTP GET request expressing preference for "application/linkset+json" response

Figure 4 shows the response to the HTTP GET request of Figure 3. The set of links is serialized according to the media type "application/linkset+json".

```
HTTP/1.1 200 OK
Date: Mon, 12 Aug 2019 10:46:22 GMT
Server: Apache-Coyote/1.1
Content-Type: application/linkset+json
Link: <https://example.org/links/resource1> ; rel="alternate"
; type="application/linkset"
Content-Length: 1349
{
  "linkset": [
    {
      "anchor": "https://example.org/resource1",
      "author": [
        {
          "href": "https://authors.example.net/johndoe",
          "type": "application/rdf+xml"
        }
      ],
      "memento": [
        {
          "href": "https://example.org/resource1?version=1",
          "type": "text/html",
          "datetime": "Thu, 13 Jun 2019 09:34:33 GMT"
        },
        {
          "href": "https://example.org/resource1?version=2",
          "type": "text/html",
          "datetime": "Sun, 21 Jul 2019 12:22:04 GMT"
        }
      ],
      "latest-version": [
        {
          "href": "https://example.org/resource1?version=3",
          "type": "text/html"
        }
      ]
    },
```

```
      {
        "anchor": "https://example.org/resource1?version=3",
        "predecessor-version": [
          {
            "href": "https://example.org/resource1?version=2",
            "type": "text/html"
          }
        ]
      },
      {
        "anchor": "https://example.org/resource1?version=2",
        "predecessor-version": [
          {
            "href": "https://example.org/resource1?version=1",
            "type": "text/html"
          }
        ]
      },
      {
        "anchor": "https://example.org/resource1#comment=1",
        "author": [
          {
            "href": "https://authors.example.net/alice"
          }
        ]
      }
    ]
  }
```

Figure 4: Response to the client's request for the set of links

7.3. Discovering a Link Set via the "linkset" Link Relation Type

Figure 5 shows a client issuing an HTTP HEAD request against resource <https://example.org/resource1>.

```
HEAD resource1 HTTP/1.1
Host: example.org
Connection: close
```

Figure 5: Client HTTP HEAD request

Figure 6 shows the response to the HEAD request of Figure 5. The response contains an HTTP "Link" header with a link that has the "linkset" relation type. It indicates that a set of links is provided by resource <https://example.org/links/resource1>, which provides a representation with media type "application/linkset+json".

```
HTTP/1.1 200 OK
Date: Mon, 12 Aug 2019 10:45:54 GMT
Server: Apache-Coyote/1.1
Link: <https://example.org/links/resource1>
      ; rel="linkset"
      ; type="application/linkset+json"
Content-Length: 236
Content-Type: text/html;charset=utf-8
Connection: close
```

Figure 6: Response to HTTP HEAD request

Section 7.2 shows a client obtaining a set of links by issuing an HTTP GET on the target of the link with the "linkset" relation type, <https://example.org/links/resource1>.

8. Implementation Status

Note to RFC Editor: Please remove this section before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 6982 [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 6982, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

8.1. GS1

GS1 is a provider of barcodes (GS1 GTINs and EAN/UPC) for retail products and manages an ecology of services and standards to leverage them at a global scale. GS1 has indicated that it will implement this "linkset" specification as a means to allow requesting and representing links pertaining to products from various retailers.

Currently, the GS1 Digital Link specification makes an informative reference to version 03 of the "linkset" I-D. GS1 expresses confidence that this will become a normative reference in the next iteration of that specification, likely to be ratified as a GS1 standard around February 2021.

8.2. FAIR Signposting Profile

The FAIR Signposting Profile is a community specification aimed at improving machine navigation of scholarly objects on the web through the use of typed web links pointing at e.g. web resources that are part of a specific object, persistent identifiers for the object and its authors, license information pertaining to the object. The specification encourages the use of Linksets and initial implementations are ongoing, for example, for the open source Dataverse data repository platform that was initiated by Harvard University and is meanwhile used by research institutions, worldwide.

8.3. Open Journal Systems (OJS)

Open Journal Systems (OJS) is an open-source software for the management of peer-reviewed academic journals, and is created by the Public Knowledge Project (PKP), released under the GNU General Public License. Open Journal Systems (OJS) is a journal management and publishing system that has been developed by PKP through its federally funded efforts to expand and improve access to research.

The OJS platform has implemented "linkset" support as an alternative way to provide links when there are more than a configured limit (they consider using about 10 as a good default, for testing purpose it is currently set to 8).

9. IANA Considerations

9.1. Link Relation Type: linkset

The link relation type below has been registered by IANA per Section 6.2.1 of Web Linking [RFC8288]:

Relation Name: linkset

Description: The Target IRI of a link with the "linkset" relation type provides a set of links, including links in which the Context IRI of the link participates.

Reference: [[This document]]

9.2. Media Type: application/linkset

The Internet media type [RFC6838] for a natively encoded linkset is application/linkset.

Type name: application

Subtype name: linkset

Required parameters: none

Optional parameters: profile

Encoding considerations: Linksets are encoded according to the definition of [RFC8288]. The encoding of [RFC8288] is based on the general encoding rules of [RFC7230], with the addition of allowing indicating character encoding and language for specific parameters as defined by [RFC8187].

Security considerations: The security considerations of [[This document]] apply.

Interoperability considerations: The interoperability considerations of [RFC7230] apply.

Published specification: [[This document]]

Applications that use this media type: This media type is not specific to any application, as it can be used by any application that wants to interchange web links.

Additional information:

Magic number(s): N/A

File extension(s): This media type does not propose a specific extension.

Macintosh file type code(s): TEXT

Person & email address to contact for further information: Erik Wilde <erik.wilde@dret.net>

Intended usage: COMMON

Restrictions on usage: none

Author: Erik Wilde <erik.wilde@dret.net>

Change controller: IETF

9.3. Media Type: application/linkset+json

The Internet media type [RFC6838] for a JSON-encoded linkset is application/linkset+json.

Type name: application

Subtype name: linkset+json

Required parameters: none

Optional parameters: profile

Encoding considerations: The encoding considerations of [RFC8259] apply

Security considerations: The security considerations of [[This document]] apply.

Interoperability considerations: The interoperability considerations of [RFC8259] apply.

Published specification: [[This document]]

Applications that use this media type: This media type is not specific to any application, as it can be used by any application that wants to interchange web links.

Additional information:

Magic number(s): N/A

File extension(s): JSON documents often use ".json" as the file extension, and this media type does not propose a specific extension other than this generic one.

Macintosh file type code(s): TEXT

Person & email address to contact for further information: Erik Wilde <erik.wilde@dret.net>

Intended usage: COMMON

Restrictions on usage: none

Author: Erik Wilde <erik.wilde@dret.net>

Change controller: IETF

10. Security Considerations

The security considerations of Web Linking [RFC8288] apply, as long as they are not specifically discussing the risks of exposing information in HTTP header fields.

In general, links may cause information leakage when they expose information (such as URIs) that can be sensitive or private. Links may expose "hidden URIs" that are not supposed to be openly shared, and may not be sufficiently protected. Ideally, none of the URIs exposed in links should be supposed to be "hidden"; instead, if these URIs are supposed to be limited to certain users, then technical measures should be put in place so that accidentally exposing them does not cause any harm.

For the specific mechanisms defined in this specification, two security considerations should be taken into account:

- The Web Linking model always has an "implicit context", which is the resource of the HTTP interaction. This original context can be lost or can change when self-contained link representations are moved. Changing the context can change the interpretation of links when they have no explicit anchor, or when they use relative URIs. Applications may choose to ignore links that have no explicit anchor or that use relative URIs when these are exchanged in stand-alone resources.

- The model introduced in this specification supports "3rd party links", where one party can provide links that have another party's resource as an anchor. Depending on the link semantics and the application context, it is important to verify that there is sufficient trust in that 3rd party to allow it to provide these links. Applications may choose to treat 3rd party links differently than cases where a resource and the links for that resource are provided by the same party.

11. References

11.1. Normative References

[RFC0822] Crocker, D., "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", STD 11, RFC 822, DOI 10.17487/RFC0822, August 1982, <https://www.rfc-editor.org/info/rfc822>.

[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/info/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

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

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/info/rfc6690>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

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

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <https://www.rfc-editor.org/info/rfc6982>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/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, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7284] Lanthaler, M., "The Profile URI Registry", RFC 7284, DOI 10.17487/RFC7284, June 2014, <https://www.rfc-editor.org/info/rfc7284>.

[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/info/rfc8174>.

[RFC8187] Reschke, J., "Indicating Character Encoding and Language for HTTP Header Field Parameters", RFC 8187, DOI 10.17487/RFC8187, September 2017, <https://www.rfc-editor.org/info/rfc8187>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

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

11.2. Informative References

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

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <https://www.rfc-editor.org/info/rfc5988>.

[W3C.REC-json-ld-20140116] Sporny, M., Kellogg, G., and M. Lanthaler, "JSON-LD 1.0", World Wide Web Consortium Recommendation REC-json-ld-20140116, January 2014, <https://www.w3.org/TR/2014/REC-json-ld-20140116>.

11.3. URIs

[1] https://www.w3.org/TR/2014/REC-json-ld-20140116/#interpreting-json-as-json-ld

[2] https://tools.ietf.org/html/rfc8288#appendix-A.2

Appendix A. Acknowledgements

Thanks for comments and suggestions provided by Phil Archer, Dominique Guinard, Mark Nottingham, Stian Soiland-Reyes, and Sarven Capadisli.

Appendix B. JSON-LD Context

A set of links rendered according to the JSON serialization defined in Section 4.2 can be interpreted as RDF triples by adding a JSON-LD context [W3C.REC-json-ld-20140116] that maps the JSON keys to corresponding Linked Data terms. And, as per [W3C.REC-json-ld-20140116] section 6.8 [1], when delivering a link set that is rendered according to the "application/linkset+json" media type to a user agent, a server can convey the availability of such a JSON-LD context by using a link with the relation type "http://www.w3.org/ns/json-ld#context" in the HTTP "Link" header.

Using the latter approach to support discovery of a JSON-LD Context, the response to the GET request of Figure 3 against the URI of a set of links would be as shown in Figure 7.

```
HTTP/1.1 200 OK
Date: Mon, 12 Aug 2019 10:48:22 GMT
Server: Apache-Coyote/1.1
Content-Type: application/linkset+json
Link: <https://example.org/contexts/linkset.jsonld>
      ; rel="http://www.w3.org/ns/json-ld#context"
      ; type="application/ld+json"
Content-Length: 846
{
  "linkset": [
    {
      "anchor": "https://example.org/article/view/7507",
      "author": [
        {
          "href": "https://orcid.org/0000-0002-1825-0097"
        }
      ],
      "item": [
        {
          "href": "https://example.org/article/7507/item/1",
          "type": "application/pdf"
        },
        {
          "href": "https://example.org/article/7507/item/2",
          "type": "text/csv"
        }
      ],
      "cite-as": [
        {
          "href": "https://doi.org/10.5555/12345680",
          "title": "A Methodology for the Emulation of Architecture"
        }
      ]
    },
    {
      "anchor": "https://example.com/links/article/7507",
      "alternate": [
        {
          "href": "https://mirror.example.com/links/article/7507",
          "type": "application/linkset"
        }
      ]
    }
  ]
}
```

Figure 7: Using a typed link to support discovery of a JSON-LD Context for a Set of Links

In order to obtain the JSON-LD Context conveyed by the server, the user agent issues an HTTP GET against the link target of the link with the "http://www.w3.org/ns/json-ld#context" relation type. The response to this GET is shown in Figure 8. This particular JSON-LD context maps "application/linkset+json" representations of link sets to Dublin Core Terms. It also renders each link relation as an absolute URI, inspired by the same approach used for Atom [RFC4287] described in [RFC8288] appendix A.2 [2].

```
HTTP/1.1 200 OK
Content-Type: application/ld+json
Content-Length: 638
{
  "@context": [
    {
      "@vocab":  "http://www.iana.org/assignments/relation/",
      "anchor":  "@id",
      "href":    "@id",
      "linkset": "@graph",
      "_linkset": "@graph",
      "title":   {
        "@id":    "http://purl.org/dc/terms/title"
      },
      "title*":  {
        "@id":    "http://purl.org/dc/terms/title"
      },
      "type":    {
        "@id":    "http://purl.org/dc/terms/format"
      }
    },
    {
      "language": "@language",
      "value":    "@value",
      "hreflang": {
        "@id":        "http://purl.org/dc/terms/language",
        "@container": "@set"
      }
    }
  ]
}
```

Figure 8: JSON-LD Context mapping to schema.org and IANA assignments

Applying the JSON-LD context of Figure 8 to the link set of Figure 7 allows transforming the "application/linkset+json" link set to an RDF link set. Figure 9 shows the latter represented by means of the "text/turtle" RDF serialization.

```
<https://example.org/article/view/7507>
    <http://www.iana.org/assignments/relation/author>
    <https://orcid.org/0000-0002-1825-0097> .
<https://example.org/article/view/7507>
    <http://www.iana.org/assignments/relation/item>
    <https://example.org/article/7507/item/1> .
<https://example.org/article/7507/item/1>
    <http://purl.org/dc/terms/format>
    "application/pdf" .
<https://example.org/article/view/7507>
    <http://www.iana.org/assignments/relation/item>
    <https://example.org/article/7507/item/2> .
<https://example.org/article/7507/item/2>
    <http://purl.org/dc/terms/format>
    "text/csv" .
<https://example.org/article/view/7507>
    <http://www.iana.org/assignments/relation/cite-as>
    <https://doi.org/10.5555/12345680> .
<https://doi.org/10.5555/12345680>
    <http://purl.org/dc/terms/title>
    "A Methodology for the Emulation of Architecture" .
<https://example.com/links/article/7507>
    <http://www.iana.org/assignments/relation/alternate>
    <https://mirror.example.com/links/article/7507> .
<https://mirror.example.com/links/article/7507>
    <http://purl.org/dc/terms/format>
    "application/linkset" .
```

Figure 9: RDF serialization of the link set resulting from applying the JSON-LD context

Note that the JSON-LD context of Figure 8 does not handle (meta)link relations of type ""linkset"" as they are in conflict with the top-level JSON key. A workaround is to rename the top-level key to ""_linkset"" in the "application/linkset+json" before transforming a link set to JSON-LD.

Authors' Addresses

Erik Wilde
Axway

Email: erik.wilde@dret.net
URI: http://dret.net/netdret/

Herbert Van de Sompel
Data Archiving and Networked Services

Email: herbert.van.de.sompel@dans.knaw.nl
URI: https://orcid.org/0000-0002-0715-6126