Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 16, 2017

A. Wright, Ed.

G. Luff

October 13, 2016

JSON Hyper-Schema: A Vocabulary for Hypermedia Annotation of JSON
draft-wright-json-schema-hyperschema-00

Abstract

JSON Schema is a JSON based format for defining the structure of JSON data. This document specifies hyperlink- and hypermedia-related keywords of JSON Schema for annotating JSON documents with hyperlinks and instructions for processing and manipulating remote JSON resources through hypermedia environments like HTTP.

Note to Readers

The issues list for this draft can be found at <https://github.com/json-schema-org/json-schema-spec/issues>.

For additional information, see <http://json-schema.org/>.

To provide feedback, use this issue tracker, the communication methods listed on the homepage, or email the document editors.

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 April 16, 2017.

Table of Contents

1. Introduction

JSON Schema is a JSON based format for defining the structure of JSON data. This document specifies hyperlink- and hypermedia-related keywords of JSON Schema.

The term JSON Hyper-Schema is used to refer to a JSON Schema that uses these keywords.

This specification will use the terminology defined by the JSON Schema core specification [json-schema]. It is advised that readers have a copy of this specification.

2. Conventions and Terminology

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 RFC 2119 [RFC2119].

The terms "schema", "instance", "property" and "item" are to be interpreted as defined in the JSON Schema core specification [json-schema].

3. Overview

This document describes how JSON Schema can be used to define hyperlinks on instance data. It also defines how to provide additional information required to interpret JSON data as rich multimedia documents.

As with all JSON Schema keywords, all the keywords described in the "Schema Keywords" section are optional. The minimal valid JSON Hyper-schema is the blank object.

Here is an example JSON Schema defining hyperlinks, and providing a multimedia interpretation for the "imgData" property:

```
{
    "title": "Written Article",
    "type": "object",
    "properties": {
        "id": {
            "title": "Article Identifier",
            "type": "number",
            "readOnly": true
        },
        "title": {
            "title": "Article Title",
            "type": "string"
        },
        "authorId": {
            "type": "integer"
        },
        "imgDataPng": {
            "title": "Article Illustration (thumbnail)",
            "type": "string",
            "media": {
                "binaryEncoding": "base64",
                "type": "image/png"
            }
        }
    },
    "required" : ["id", "title", "authorId"],
    "links": [
        {
            "rel": "self",
            "href": "/article{?id}"
        },
        {
            "rel": "author",
            "href": "/user?id={authorId}"
        }
    ]
}
```

This example schema defines the properties of the instance. For the "imgData" property, it specifies that that it should be base64-decoded and the resulting binary data treated as a PNG image. It also defines link relations for the instance, with URIs incorporating values from the instance. [[CREF1: "id" probably should not normally be a required keyword, since new instances will have an unknown "id" property until is it assigned by the server. However, this property is used in a link, and without it, multiple different instances would be given the same rel=self URI!]]

An example of a JSON instance described by the above schema might be:

```
{
    "id": 15,
    "title": "Example data",
    "authorId": 105,
    "imgData": "iVBORw...kJggg=="
}
```

The base-64 data has been abbreviated for readability.

4. Schema keywords

4.1. base

If present, this keyword is resolved against the current URI base that the entire instance is found within, and sets the new URI base for URI references within the instance. It is therefore the first URI Reference resolved, regardless of which order it was found in.

The URI is computed from the provided URI template using the same process described for the "href" (Section 5.1) property of a Link Description Object.

An example of a JSON schema using "base":

```
{
    "base": "/object/{id}",
    "links": [
        {
            "rel": "self",
            "href": ""
        },
        {
            "rel": "next",
            "href": "{next_id}"
        }
    ]
}
```

An example of a JSON instance using this schema to produce rel="self" and rel="next" links:

```
{
    "id": "41",
    "next_id": "42"
}
```

If the document URI is <http://example.com/?id=41>, then the new URI base becomes <http://example.com/object/41>

Resolving the two Link Description Objects against this URI base creates two links exactly equivelant to these absolute-form HTTP Link headers:

- Link: <http://example.com/object/41>;rel=self

- Link: <http://example.com/object/42>;rel=next

4.2. links

The "links" property of schemas is used to associate Link Description Objects with instances. The value of this property MUST be an array, and the items in the array must be Link Description Objects, as defined below.

An example schema using the "links" keyword could be:

```
{
    "title": "Schema defining links",
    "links": [
        {
            "rel": "self",
            "href": "{id}"
        },
        {
            "rel": "parent",
            "href": "{parent}"
        }
    ]
}
```

4.3. media

The "media" property indicates that this instance contains non-JSON data encoded in a JSON string. It describes the type of content and how it is encoded.

The value of this property MUST be an object. The value of this property SHOULD be ignored if the instance described is not a string.

4.3.1. Properties of "media"

The value of the "media" keyword MAY contain any of the following properties:

4.3.1.1. binaryEncoding

If the instance value is a string, this property defines that the string SHOULD be interpreted as binary data and decoded using the encoding named by this property. RFC 2045, Sec 6.1 [RFC2045] lists the possible values for this property.

4.3.1.2. type

The value of this property must be a media type, as defined by RFC 2046 [RFC2046]. This property defines the media type of instances which this schema defines.

If the "binaryEncoding" property is not set, but the instance value is a string, then the value of this property SHOULD specify a text document type, and the character set SHOULD be the character set into which the JSON string value was decoded (for which the default is Unicode).

4.3.2. Example

Here is an example schema, illustrating the use of "media":

```
{
    "type": "string",
    "media": {
        "binaryEncoding": "base64",
        "type": "image/png"
    }
}
```

Instances described by this schema should be strings, and their values should be interpretable as base64-encoded PNG images.

Another example:

```
{
    "type": "string",
    "media": {
        "mediaType": "text/html"
    }
}
```

Instances described by this schema should be strings containing HTML, using whatever character set the JSON string was decoded into (default is Unicode).

4.4. readOnly

If it has a value of boolean true, this keyword indicates that the value of the instance is managed exclusively by the server or the owning authority, and attempts by a user agent to modify the value of this property are expected to be ignored or rejected by a server.

For example, this property would be used to mark a server-generated serial number as read-only.

The value of this keyword MUST be a boolean. The default value is false.

5. Link Description Object

A Link Description Object (LDO) is used to describe a single link relation from the instance to another resource. A Link Description Object must be an object.

The link description format can be used without JSON Schema, and use of this format can be declared by referencing the normative link description schema as the schema for the data structure that uses the links. The URI of the normative link description schema is: http://json-schema.org/draft-04/links (draft-04 version).

"Form"-like functionality can be defined by use of the "method" and "schema" keywords, which supplies a schema describing the data to supply to the server.

5.1. href

The value of the "href" link description property is a template used to determine the target URI of the related resource. The value of the instance property MUST be resolved as a URI-reference [RFC3986] against the base URI of the instance.

This property is REQUIRED.

5.1.1. URI Templating

The value of "href" is to be used as a URI Template, as defined in RFC 6570 [RFC6570]. However, some special considerations apply:

5.1.1.1. Pre-processing

[[CREF2: This pre-processing section is subject to significant change in upcoming drafts.]]

The URI Template specification [RFC6570] restricts the set of characters available for variable names. Property names in JSON, however, can be any UTF-8 string.

To allow the use of any JSON property name in the template, before using the value of "href" as a URI Template, the following pre-processing rules MUST be applied, in order:

5.1.1.1.1. Bracket escaping

The purpose of this step is to allow the use of brackets to percent-encode variable names inside curly brackets. Variable names to be escaped are enclosed within rounded brackets, with the close-rounded-bracket character ")" being escaped as a pair of close-rounded-brackets "))". Since the empty string is not a valid variable name in RFC 6570, an empty pair of brackets is replaced with "%65mpty".

The rules are as follows:

Find the largest possible sections of the text such that:

do not contain an odd number of close-rounded-bracket characters ")" in sequence in that section of the text

are surrounded by a pair of rounded brackets: ( ), where

the surrounding rounded brackets are themselves contained within a pair of curly brackets: { }

Each of these sections of the text (including the surrounding rounded brackets) MUST be replaced, according to the following rules:

If the brackets contained no text (the empty string), then they are replaced with "%65mpty" (which is "empty" with a percent-encoded "e")

Otherwise, the enclosing brackets are removed, and the inner text used after the following modifications

all pairs of close-brackets "))" are replaced with a single close bracket

after that, the text is replaced with its percent-encoded equivalent, such that the result is a valid RFC 6570 variable name (note that this requires encoding characters such as "*" and "!")

5.1.1.1.2. Replacing $

After the above substitutions, if the character "$" (dollar sign) appears within a pair of curly brackets, then it MUST be replaced with the text "%73elf" (which is "self" with a percent-encoded "s").

The purpose of this stage is to allow the use of the instance value itself (instead of its object properties or array items) in the URI Template, by the special value "%73elf".

5.1.1.1.3. Choice of special-case values

The special-case values of "%73elf" and "%65mpty" were chosen because they are unlikely to be accidentally generated by either a human or automated escaping.

5.1.1.1.4. Examples

For example, here are some possible values for "href", followed by the results after pre-processing:

Input	Output
"no change"	"no change"
"(no change)"	"(no change)"
"{(escape space)}"	"{escape%20space}"
"{(escape+plus)}"	"{escape%2Bplus}"
"{(escape*asterisk)}"	"{escape%2Aasterisk}"
"{(escape(bracket)}"	"{escape%28bracket}"
"{(escape))bracket)}"	"{escape%29bracket}"
"{(a))b)}"	"{a%29b}
"{(a (b)))}"	"{a%20%28b%29}
"{()}"	"{%65mpty}
"{+$*}"	"{+%73elf*}
"{+($)*}"	"{+%24*}

Note that in the final example, because the "+" was outside the brackets, it remained unescaped, whereas in the fourth example the "+" was escaped.

5.1.1.2. Values for substitution

After pre-processing, the URI Template is filled out using data from the instance. To allow the use of any object property (including the empty string), array index, or the instance value itself, the following rules are defined:

For a given variable name in the URI Template, the value to use is determined as follows:

If the variable name is "%73elf", then the instance value itself MUST be used.

If the variable name is "%65mpty", then the instances's empty-string ("") property MUST be used (if it exists).

If the instance is an array, and the variable name is a representation of a non-negative integer, then the value at the corresponding array index MUST be used (if it exists).

Otherwise, the variable name should be percent-decoded, and the corresponding object property MUST be used (if it exists).

5.1.1.2.1. Converting to strings

When any value referenced by the URI template is null, a boolean or a number, then it should first be converted into a string as follows:

null values SHOULD be replaced by the text "null"

boolean values SHOULD be replaced by their lower-case equivalents: "true" or "false"

numbers SHOULD be replaced with their original JSON representation.

In some software environments the original JSON representation of a number will not be available (there is no way to tell the difference between 1.0 and 1), so any reasonable representation should be used. Schema and API authors should bear this in mind, and use other types (such as string or boolean) if the exact representation is important.

5.1.1.3. Missing values

Sometimes, the appropriate values will not be available. For example, the template might specify the use of object properties, but the instance is an array or a string.

If any of the values required for the template are not present in the JSON instance, then substitute values MAY be provided from another source (such as default values). Otherwise, the link definition SHOULD be considered not to apply to the instance.

5.2. rel

The value of the "rel" property indicates the name of the relation to the target resource. The value MUST be a registered link relation from the IANA Link Relation Type Registry established in RFC 5988 [RFC5988], or a normalized URI following the URI production of RFC 3986 [RFC3986].

The relation to the target is interpreted as from the instance that the schema (or sub-schema) applies to, not any larger document that the instance may have been found in.

Relationship definitions are not normally media type dependent, and users are encouraged to utilize existing accepted relation definitions.

For example, if a schema is defined:

```
{
    "links": [{
        "rel": "self",
        "href": "{id}"
    }, {
        "rel": "up",
        "href": "{upId}"
    }]
}
```

And if a collection of instance resources were retrieved with JSON representation:

```
GET /Resource/

[{
    "id": "thing",
    "upId": "parent"
}, {
    "id": "thing2",
    "upId": "parent"
}]
```

This would indicate that for the first item in the collection, its own (self) URI would resolve to "/Resource/thing" and the first item's "up" relation SHOULD be resolved to the resource at "/Resource/parent".

Note that these relationship values are case-insensitive, consistent with their use in HTML and the HTTP Link header [RFC5988].

5.2.1. Security Considerations for "self" links

When link relation of "self" is used to denote a full representation of an object, the user agent SHOULD NOT consider the representation to be the authoritative representation of the resource denoted by the target URI if the target URI is not equivalent to or a sub-path of the the URI used to request the resource representation which contains the target URI with the "self" link.

For example, if a hyper schema was defined:

```
{
    "links": [{
        "rel": "self",
        "href": "{id}"
    }]
}
```

And a resource was requested from somesite.com:

```
GET /foo/
```

With a response of (with newlines and whitespace added):

```
Content-Type: application/json; profile="http://example.com/alpha"

[{
    "id": "bar",
    "name": "This representation can be safely treated
             as authoritative "
}, {
    "id": "/baz",
    "name": "This representation should not be treated as
             authoritative the user agent should make request the
             resource from '/baz' to ensure it has the authoritative
             representation"
}, {
    "id": "http://othersite.com/something",
    "name": "This representation
             should also not be treated as authoritative and the
             target resource representation should be retrieved
             for the authoritative representation"
}]
```

5.3. title

This property defines a title for the link. The value must be a string.

User agents MAY use this title when presenting the link to the user.

5.4. targetSchema

This property provides a schema that is expected to describe the link target, including what a client can expect if it makes an HTTP GET request, and what it should send if it replaces the resource in an HTTP PUT request. This property is advisory only.

5.4.1. Security Considerations for "targetSchema"

This property has similar security concerns to that of "mediaType". Clients MUST NOT use the value of this property to aid in the interpretation of the data received in response to following the link, as this leaves "safe" data open to re-interpretation.

For example, suppose two programmers are having a discussion about web security using a text-only message board. Here is some data from that conversation, with a URI of: http://forum.example.com/topics/152/comments/13

```
{
    "topicId": 152,
    "commentId": 13,
    "from": {
        "name": "Jane",
        "id": 5
    },
    "to": {
        "name": "Jason",
        "id": 8
    },
    "message": "It's easy, just add some HTML like
         this: <script>doSomethingEvil()</script>"
}
```

The message string was split over two lines for readability.

A third party might then write provide the following Link Description Object at another location:

```
{
    "rel": "evil-attack",
    "href": "http://forum.example.com/topics/152/comments/13",
    "targetSchema": {
        "properties": {
            "message": {
                "description": "Re-interpret `message` as HTML",
                "media": {
                    "type": "text/html"
                }
            }
        }
    }
}
```

If the client used this "targetSchema" value when interpreting the above data, then it might display the contents of "message" as HTML. At this point, the JavaScript embedded in the message might be executed (in the context of the "forum.example.com" domain).

5.5. mediaType

The value of this property is advisory only, and represents the media type RFC 2046 [RFC2046], that is expected to be returned when fetching this resource. This property value MAY be a media range instead, using the same pattern defined in RFC 7231, section 5.3.1 - HTTP "Accept" header [RFC7231].

This property is analogous to the "type" property of <a> elements in HTML (advisory content type), or the "type" parameter in the HTTP Link header [RFC5988]. User agents MAY use this information to inform the interface they present to the user before the link is followed, but this information MUST NOT use this information in the interpretation of the resulting data. When deciding how to interpret data obtained through following this link, the behaviour of user agents MUST be identical regardless of the value of the this property.

If this property's value is specified, and the link's target is to be obtained using any protocol that supports the HTTP/1.1 "Accept" header RFC 7231, section 5.3.1 [RFC7231], then user agents MAY use the value of this property to aid in the assembly of that header when making the request to the server.

If this property's value is not specified, then the value should be taken to be "application/json".

For example, if a schema is defined:

```
{
    "links": [{
        "rel": "self",
        "href": "/{id}/json"
    }, {
        "rel": "alternate",
        "href": "/{id}/html",
        "mediaType": "text/html"
    }, {
        "rel": "alternate",
        "href": "/{id}/rss",
        "mediaType": "application/rss+xml"
    }, {
        "rel": "icon",
        "href": "{id}/icon",
        "mediaType": "image/*"
    }]
}
```

A suitable instance described by this schema would have four links defined. The link with a "rel" value of "self" would have an expected MIME type of "application/json" (the default). The two links with a "rel" value of "alternate" specify the locations of HTML and RSS versions of the current item. The link with a "rel" value of "icon" links to an image, but does not specify the exact format.

A visual user agent displaying the item from the above example might present a button representing an RSS feed, which when pressed passes the target URI (calculated "href" value) to an view more suited to displaying it, such as a news feed aggregator tab.

Note that presenting the link in the above manner, or passing the URI to a news feed aggregator view does not constitute interpretation of the data, but an interpretation of the link. The interpretation of the data itself is performed by the news feed aggregator, which SHOULD reject any data that would not have also been interpreted as a news feed, had it been displayed in the main view.

5.5.1. Security concerns for "mediaType"

The "mediaType" property in link definitions defines the expected format of the link's target. However, this is advisory only, and MUST NOT be considered authoritative.

When choosing how to interpret data, the type information provided by the server (or inferred from the filename, or any other usual method) MUST be the only consideration, and the "mediaType" property of the link MUST NOT be used. User agents MAY use this information to determine how they represent the link or where to display it (for example hover-text, opening in a new tab). If user agents decide to pass the link to an external program, they SHOULD first verify that the data is of a type that would normally be passed to that external program.

This is to guard against re-interpretation of "safe" data, similar to the precautions for "targetSchema".

5.6. Submission Form Properties

The following properties also apply to Link Description Objects, and provide functionality analogous to HTMLforms [W3C.CR-html5-20140731], by providing a means for making a request with client- or user-selected information.

5.6.1. method

This property specifies that the client can construct a templated query or non-idempotent request to a resource.

If "method" is "get", the link identifies how a user can compute the URI of an arbitrary resource. For example, how compute a link to a page of search results relating to the instance, for a user-selected query term. Despite being named after GET, there is no constraint on the method or protocol used to interact with the remote resource.

If "method" is "post", the link specifies how a user can construct a document to submit to the link target for evaluation.

Values for this property SHOULD be lowercase, and SHOULD be compared case-insensitive. Use of other values not defined here SHOULD be ignored.

5.6.2. encType

If present, this property indicates the media type format the client should use to encode a query parameter or send to the server. posting to the collection of instances at the target resource. If the method is "get", this will indicate how to encode the query-string that is appended to the "href" link target. If the method is "post", this indicates which media type to send to the server and how to encode it.

For example, with the following schema:

```
{
    "links": [{
        "encType": "application/x-www-form-urlencoded",
        "method": "get",
        "href": "/Product/",
        "properties": {
            "name": {
                "description": "name of the product"
            }
        }
    }]
}
```

This indicates that the client can query the server for instances that have a specific name.

For example:

```
/Product/?name=Slinky
```

If the method is "post", "application/json" is the default media type.

5.6.3. schema

This property contains a schema which defines the acceptable structure of the document being encoded according to the "encType" property.

Note that this does not provide data for any URI templates. This is a separate concept from the "targetSchema" property, which is describing the target information resource (including for replacing the contents of the resource in a PUT request), unlike "schema" which describes the user-submitted request data to be evaluated by the resource.

6. References

6.1. Normative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <http://www.rfc-editor.org/info/rfc2045>.

[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>.

[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>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <http://www.rfc-editor.org/info/rfc6570>.

[json-schema] Wright, A., "JSON Schema: A Media Type for Describing JSON Documents", draft-wright-json-schema-00 (work in progress), October 2016.

6.2. 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>.

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

[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>.

[W3C.CR-html5-20140731] Berjon, R., Faulkner, S., Leithead, T., Navara, E., O'Connor, E., and S. Pfeiffer, "HTML5", World Wide Web Consortium CR CR-html5-20140731, July 2014, <http://www.w3.org/TR/2014/CR-html5-20140731>.

Appendix A. Acknowledgments

Thanks to Gary Court, Francis Galiegue, Kris Zyp, and Geraint Luff for their work on the initial drafts of JSON Schema.

Thanks to Jason Desrosiers, Daniel Perrett, Erik Wilde, Ben Hutton, Evgeny Poberezkin, and Henry H. Andrews for their submissions and patches to the document.

Appendix B. Change Log

[[CREF3: This section to be removed before leaving Internet-Draft status.]]

draft-wright-json-schema-hyperschema-00

* "rel" is now optional
* rel="self" no longer changes URI base
* Added "base" keyword to change instance URI base
* Removed "root" link relation
* Removed "create" link relation
* Removed "full" link relation
* Removed "instances" link relation
* Removed special behavior for "describedBy" link relation
* Removed "pathStart" keyword
* Removed "fragmentResolution" keyword
* Updated references to JSON Pointer, HTML
* Changed behavior of "method" property to align with hypermedia best current practices

draft-luff-json-hyper-schema-01

* Split from main specification.

Authors' Addresses

Austin Wright (editor)

EMail: aaa@bzfx.net

Geraint Luff
Cambridge
UK

EMail: luffgd@gmail.com