Internet Engineering Task Force A. Wright, Ed.
Internet-Draft
Intended status: Informational H. Andrews, Ed.
Expires: October 17, 2017 Cloudflare, Inc.
G. Luff
April 15, 2017

JSON Hyper-Schema: A Vocabulary for Hypermedia Annotation of JSON
draft-wright-json-schema-hyperschema-01

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

Copyright Notice

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" and "instance" 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"
           },
           "imgData": {
               "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.

3.1. Interaction with validation

Hyper-schemas MUST NOT be applied to an instance if the instance fails to validate against the validation keywords within or containing the hyper-schema. Hyper-schema keywords in branches of an "anyOf" or "oneOf" that do not validate, or in a "dependencies" subschema that is not relevant to the instance, MUST be ignored.

Hyper-schema keywords in a subschema contained within a "not", at any depth, including any number of intervening additional "not" subschemas, MUST be ignored.

If the subschema for a "contains" keyword contains hyper-schema keywords they MUST be applied to all array elements that validate against the schema. While finding a single validating element is sufficient to determine the validation outcome, when hyper-schema keywords are present, the subschema MUST be evaluated against all array elements.

4. Meta-schema

The current URI for the JSON Schema Validation is <http://json-schema.org/draft-06/hyper-schema#>.

5. Schema keywords

5.1. 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 6.2) property of a Link Description Object.

An example of a JSON schema using "base":

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

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

```
{
    "id": 41,
    "nextId": 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 equivalent to these absolute-form HTTP Link headers:

- Link: <http://example.com/object/41>;rel=self
- Link: <http://example.com/object/42>;rel=next

5.2. 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}"
        }
    ]
}
```

5.3. 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.

5.3.1. Properties of "media"

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

5.3.1.1. 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.

5.3.1.2. 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).

5.3.2. 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": {
        "type": "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).

5.4. 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.

6. 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-06/links (draft-06 version).

6.1. Links, operations, and data

[[CREF2: Note that while the current draft does not provide a way to explicity indicate HTTP method support, some way of providing a non-authoritative hint may be added in a future draft (see issue #73 in the GitHub repository).]]

There are several ways that a client can use data can with a link:

URI Template variables resolved from server-supplied instance data

URI Template variables resolved from user agent data

Replacing or modifying the target resource's representation

Submitting data for processing, where the data has no inherent relation to the target resource's representation

The three ways to use client-supplied data are each addressed by a separate schema keyword within the link description object. Link operations ignore schemas that are not relevant to their semantics.

Link Description Objects do not directly indicate what operations, such as HTTP methods, are supported by the target resource. Instead, operations should be inferred primarily from link relation types (Section 6.4) and URI schemes. Note, however, that a resource may always decline an operation at runtime, for instance due to application state that controls the operation's availability.

6.1.1. Resolving templated URIs

URI Template variables in "href" (Section 6.2) resolve from server-supplied instance data by default. "hrefSchema" (Section 6.3) allows a link to specify a schema for resolving template variables from client-supplied data. Regular JSON Schema validation features can be used to require resolution from user agent data, forbid it, or allow

user agent data while falling back to server-supplied instance data if no user agent data is provided.

The common pattern of resolving a templated path component with server-supplied instance data while accepting user agent data to build a query string can be implemented by setting the "hrefSchema" subschemas for the path template variables to false, while giving the query string template variables names that do not appear in the instance. This ensures that the path variables can only be resolved from the instance, and the query string variables can only be resolved from user agent data. See the "hrefSchema" section for an example of this approach.

6.1.2. Manipulating the target resource representation

In JSON Hyper-Schema, "targetSchema" (Section 6.6) supplies a non-authoritative description of the target resource's representation. A client can use "targetSchema" to structure input for replacing or modifying the representation. Alternatively, if "targetSchema" is absent or if the client prefers to only use authoritative information, it can interact with the target resource to confirm or discover its representation structure.

"targetSchema" is not intended to describe link operation responses, except when the response semantics indicate that it is a representation of the target resource. In all cases, the schema indicated by the response itself is authoritative. See the Section 6.6.1 for guidance specific to each HTTP method when using "targetSchema" with HTTP URIs.

6.1.3. Submitting data for processing

The "submissionSchema" (Section 6.9) and "submissionEncType" (Section 6.8) keywords describe the domain of the processing function implemented by the target resource. Otherwise, as noted above, the submission schema and encoding are ignored for operations to which they are not relevant.

6.2. 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.

6.2.1. URI Templating

[[CREF3: The pre-processing rules present in earlier drafts have been removed due to their complexity and inability to address all limitations with URI templating. This section is subject to significant change in upcoming drafts to replace the old pre-processing with a comprehensive solution.]]

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

6.2.1.1. Values for substitution

The URI Template is filled out using data from some combination of an external source and the instance. Where either instance data or user agent data may be used, this section will refer simply to "data" or to a "value". When the source is important, it is specified explicitly. To allow the use of any object property (including the empty string) or array index, the following rules are defined:

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

If the data 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).

If "hrefSchema" (Section 6.3) is present and user agent data is provided, the data MUST be a valid instance according to the value of "hrefSchema". Template variables, after the process listed above, MUST first be resolved from the user agent data instance. Any variables left unresolved MUST be resolved from the resource instance data.

6.2.1.1.1. 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.

6.2.1.2. Missing values

Sometimes, the appropriate values will not be available. For example, the template might specify the use of object properties, but no such data was provided (or "hrefSchema" is not present), and the instance is an array or a string.

If any of the values required for the template are neither present in the user agent data (if relevant) nor 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.

6.3. hrefSchema

The value of the "hrefSchema" link description property MUST be a valid JSON Schema. This schema is used to validate user input or other user agent data for filling out the URI Template in "href" (Section 6.2), as described in that section.

Omitting "hrefSchema" or setting the entire schema to "false" prevents any user agent data from being accepted.

Implementations MUST NOT attempt to validate values resolved from resource instance data with "hrefSchema". This allows for different validation rules for user agent data, such as supporting spelled-out months for date-time input but using the standard date-time format for storage.

For example, this defines a schema for each of the query string parameters in the URI template:

```
{
    "href": "/foos{?condition,count,query}",
    "hrefSchema": {
        "properties": {
            "condition": {
                "type": "boolean",
                "default": true
            },
            "count": {
                "type": "integer",
                "minimum": 0,
                "default": 0
            },
            "query": {
                "type": "string"
            }
        }
    }
}
```

In this example, the schema for "extra" is given as a reference to keep the user agent data validation constraints identical to the instance validation constraints for the corresponding property, while "id" is given a false schema to prevent user agent data for that variable.

```
{
    "definitions": {
        "extra": {
            "type": "string",
            "maxLength": 32
        }
    },
    "type": "object",
    "properties": {
        "id": {
            "type": "integer",
            "minimum": 1,
            "readOnly": true
        },
        "extra": {"$ref": "#/definitions/extra"}
    },
    "links": [{
        "rel": "self",
        "href": "/things/{id}{?extra}",
        "hrefSchema": {
            "properties": {
                "id": false,
                "extra": {"$ref": "#/definitions/extra"}
            }
        }
    }]
}
```

[[CREF4: The above example simulates the behavior handled in earlier drafts with a "method" of "get" by using the new "hrefSchema" keyword.]]

6.4. 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 hyper-schema is defined:

```
{
    "type": "array",
    "items": {
        "links": [{
            "rel": "item",
            "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 URI as its own resource 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].

6.4.1. 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 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"
}]
```

6.5. 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.

6.6. targetSchema

This property provides a schema that is expected to describe the link target's representation. Depending on the protocol, the schema may or may not describe the response to any particular request sent to the link. This property is advisory only.

6.6.1. "targetSchema" and HTTP

The relationship between a resource's representation and HTTP requests and responses is determined by RFC 7231, section 4.3.1 - "GET", section 4.3.4 "PUT", and section 3.1.4.2, "Content-Location" [RFC7231]. In particular, "targetSchema" suggests what a client can expect for the response to an HTTP GET or any response for which the "Content-Location" header is equal to the request URI, and what a client should send if it replaces the resource in an HTTP PUT request. Per RFC 5789 [RFC5789], the request structure for an HTTP PATCH is determined by the combination of "targetSchema" and the request media type.

6.6.2. 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 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).

6.7. 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 5.3.2 - 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 5.3.2 [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.

6.7.1. 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".

6.8. submissionEncType

If present, this property indicates the media type format the client should use for the request payload described by "submissionSchema" (Section 6.9).

Omitting this keyword has the same behavior as a value of application/json.

Note that "submissionEncType" and "submissionSchema" are not restricted to HTTP URIs.

For example, this link indicates that if you want to send an email to the author of the context resource, your client needs to ask for both a plain text and an HTML representation.

```
{
    "links": [{
        "submissionEncType": "multipart/alternative; boundary=ab12",
        "rel": "author",
        "href": "mailto:someone@example.com{?subject}",
        "hrefSchema": {
            "type": "object",
            "properties": {
                "subject": { "type": "string" }
            },
            "required": ["subject"]
        },
        "submissionSchema": {
            "type": "array",
            "items": [
                {
                    "type": "string",
                    "media": { "type": "text/plain; charset=utf8" }
                },
                {
                    "type": "string",
                    "media": { "type": "text/html" }
                }
            ],
            "minItems": 2
        }
    }]
}
```

6.9. submissionSchema

This property contains a schema which defines the acceptable structure of the document to be encoded according to the "submissionEncType" property and sent to the target resource for processing. This can be viewed as describing the domain of the processing function implemented by the target resource.

This is a separate concept from the "targetSchema" (Section 6.6) property, which is describing the target information resource (including for replacing the contents of the resource in a PUT request), unlike "submissionSchema" which describes the user-submitted request data to be evaluated by the resource. "submissionSchema" is intended for use with requests that have payloads that are not defined in terms of the target representation.

Omitting "submissionSchema" or setting the entire schema to "false" prevents any user agent data from being accepted.

7. References

7.1. 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.

7.2. Informative References

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

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

[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>.

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, Brad Bowman, Gowry Sankar, Donald Pipowitch, Dave Finlay, and Denis Laxalde for their submissions and patches to the document.

Appendix B. Change Log

[[CREF5: This section to be removed before leaving Internet-Draft status.]]

draft-wright-json-schema-hyperschema-01

* Fixed examples
* Added "hrefSchema" for user input to "href" URI Templates
* Removed URI Template pre-processing
* Clarified how links and data submission work
* Clarified how validation keywords apply hyper-schema keywords and links
* Clarified HTTP use with "targetSchema"
* Renamed "schema" to "submissionSchema"
* Renamed "encType" to "submissionEncType"
* Removed "method"

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

Henry Andrews (editor)
Cloudflare, Inc.

EMail: henry@cloudflare.com

Geraint Luff
Cambridge
UK

EMail: luffgd@gmail.com