Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 5, 2013

G. Luff, Ed.
K. Zyp
SitePen (USA)
G. Court

February 1, 2013

JSON Hyper-Schema: Hypertext definitions for JSON Schema
draft-luff-json-hyper-schema-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.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 5, 2013.

Copyright Notice

Table of Contents

1. Introduction 3
2. Conventions and Terminology 3
3. Overview 3
3.1. Design Considerations 5
4. Schema keywords 5
4.1. links 5
4.1.1. Multiple links per URI 6
4.2. fragmentResolution 8
4.2.1. json-pointer fragment resolution 8
4.3. media 8
4.3.1. Properties of "media" 9
4.3.2. Example 9
4.4. readOnly 10
4.5. pathStart 10
5. Link Description Object 10
5.1. href 11
5.1.1. URI Templating 11
5.2. rel 14
5.2.1. Fragment resolution with "root" links 16
5.2.2. Security Considerations for "self" links 17
5.3. title 18
5.4. targetSchema 18
5.4.1. Security Considerations for "targetSchema" 19
5.5. mediaType 20
5.5.1. Security concerns for "mediaType" 21
5.6. Submission Link Properties 22
5.6.1. method 22
5.6.2. encType 22
5.6.3. schema 23
6. IANA Considerations 23
6.1. Registry of Link Relations 23
7. References 24
7.1. Normative References 24
7.2. Informative References 24
Appendix A. Change Log 25

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-core]. 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-core].

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.

Just as with the core JSON schema keywords, all the keywords described in the "Schema Keywords" section are optional.

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"
        },
        "title": {
            "title": "Article Title",
            "type": "string"
        },
        "authorId": {
            "type": "integer"
        },
        "imgData": {
            "title": "Article Illustration (small)",
            "type": "string",
            "media": {
                "binaryEncoding": "base64",
                "type": "image/png"
            }
        }
    },
    "required" : ["id", "title", "authorId"],
    "links": [
        {
            "rel": "full",
            "href": "{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.

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. Design Considerations

The purpose of this document is to define keywords for the JSON Schema that allow JSON data to be understood as hyper-text.

JSON data on its own requires special knowledge from the client about the format in order to be interpretable as hyper-text. This document proposes a way to describe the hyper-text and hyper-media interpretation of such JSON formats, without defining reserved keywords or otherwise restricting the structure of the JSON data.

4. Schema keywords

4.1. 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": "full",
            "href": "{id}"
        },
        {
            "rel": "parent",
            "href": "{parent}"
        }
    ]
}
```

4.1.1. Multiple links per URI

A single URI might have more than one role with relation to an instance. This is not a problem - the same URI can be used in more than one Link Description Object.

For example, this schema describes a format for blog posts, accessed via HTTP. The links describe how to access the comments for the post, how to search the comments, and how to submit new comments, all with the same URI:

```
{
    "title": "News post",
    ...
    "links": [
        {
            "rel": "comments",
            "href": "/{id}/comments"
        },
        {
            "rel": "search",
            "href": "/{id}/comments",
            "schema": {
                "type": "object",
                "properties": {
                    "searchTerm": {
                        "type": "string"
                    },
                    "itemsPerPage": {
                        "type": "integer",
                        "minimum": 10,
                        "multipleOf": 10,
                        "default": 20
                    }
                },
                "required": ["searchTerm"]
            }
        },
        {
            "title": "Post a comment",
            "rel": "create",
            "href": "/{id}/comments",
            "method": "POST",
            "schema": {
                "type": "object",
                "properties": {
                    "message": {
                        "type": "string"
                    }
                },
                "required": ["message"]
            }
        }
    ]
```

```
}
```

If the client follows the first link, the URI might be expanded to "/15/comments". For the second link, the method is "GET" (the default for HTTP) so a client following this link would add the parameters to the URL to produce something like: "/15/comments?searchTerm=JSON&itemsPerPage=50". The third link defines a possible interaction where a client would POST to a URI (such as "/15/comments"), where the post-data was a JSON representation of the new comment, for example:

```
{
    "message": "This is an example comment"
}
```

4.2. fragmentResolution

When addressing a JSON document, the fragment part of the URI may be used to refer to a particular instance within the document.

This keyword indicates the method to use for finding the appropriate instance within a document, given the fragment part. The default fragment resolution protocol is "json-pointer", which is defined below. Other fragment resolution protocols MAY be used, but are not defined in this document.

If the instance is described by a schema providing the a link with "root" relation, or such a link is provided in using the HTTP Link header [RFC5988], then the target of the "root" link should be considered the document root for the purposes of all fragment resolution methods that use the document structure (such as "json-pointer"). The only exception to this is the resolution of "root" links themselves.

4.2.1. json-pointer fragment resolution

The "json-pointer" fragment resolution protocol uses a JSON Pointer [json-pointer] to resolve fragment identifiers in URIs within instance representations.

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, and SHOULD be ignored for any instance that 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 instance property SHOULD NOT be changed, and attempts by a user agent to modify the value of this property are expected to be rejected by a server.

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

4.5. pathStart

This property is a URI that defines what the instance's URI MUST start with in order to validate. The value of the "pathStart" property MUST be resolved relative to the closest URI Resolution Scope (as defined in the JSON Schema core specification [json-schema-core]), using the rules from RFC 3986, Sec 5 [RFC3986].

When multiple schemas have been referenced for an instance, the user agent can determine if this schema is applicable for a particular instance by determining if the URI of the instance begins with the the value of the "pathStart" property. If the URI of the instance does not start with this URI, or if another schema specifies a starting URI that is longer and also matches the instance, this schema SHOULD NOT be considered to describe the instance. Any schema that does not have a pathStart property SHOULD be considered applicable to all the instances for which it is referenced.

5. Link Description Object

A Link Description Object (LDO) is used to describe a single link relation. In the context of a schema, it defines the link relations of the instances of the schema, and can be parameterized by the

instance values. A Link Description Object (LDO) 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/links (latest version) or http://json-schema.org/draft-04/links (draft-04 version).

"Form"-like functionality can be defined by use of the "schema" keyword, 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 SHOULD be resolved as a URI-Reference per RFC 3986 [RFC3986] and MAY be a relative reference to a URI. The base URI to be used for relative URI resolution SHOULD be the URI used to retrieve the instance object (not the schema).

The base URI to be used for relative URI resolution SHOULD is defined as follows:

if the data has a link defined, with a relation of "self", then the "href" value of that link is used, unless the relation of the link being resolved is also "self"

otherwise, the URI should be resolved against the link with relation "self" belonging to the closest parent node in the JSON document, if it exists

otherwise, the URI used to fetch the document should be used.

This property is not optional.

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

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. This property is not optional.

The relation to the target SHOULD be interpreted as specifically from the instance object that the schema (or sub-schema) applies to, not just the top level resource that contains the object within its

hierarchy. A link relation from the top level resource to a target MUST be indicated with the schema describing the top level JSON representation.

Relationship definitions SHOULD NOT be media type dependent, and users are encouraged to utilize existing accepted relation definitions, including those in existing relation registries (see RFC 4287 [RFC4287]). However, we define these relations here for clarity of normative interpretation within the context of JSON Schema defined relations:

self If the relation value is "self", when this property is encountered in the instance object, the object represents a resource and the instance object is treated as a full representation of the target resource identified by the specified URI.

full This indicates that the target of the link is the full representation for the instance object. The instance that contains this link may not be the full representation.

describedBy This indicates the target of the link is a schema describing the instance object. This MAY be used to specifically denote the schemas of objects within a JSON object hierarchy, facilitating polymorphic type data structures.

root This relation indicates that the target of the link SHOULD be treated as the root or the body of the representation for the purposes of user agent interaction or fragment resolution. All other data in the document can be regarded as meta-data for the document. The URI of this link MUST refer to a location within the instance document, otherwise the link MUST be ignored.

The following relations are applicable for schemas (the schema as the "from" resource in the relation) if they require no parameterization with data from the instance:

instances This indicates the target resource that represents a collection of instances of a schema.

create This indicates a target to use for creating new instances of a schema. This link definition SHOULD be a submission link with a non-safe method (like POST).

For example, if a schema is defined:

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

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". The "children" collection would be located at "/Resource/?upId=thing".

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

5.2.1. Fragment resolution with "root" links

The presence of a link with relation "root" alters what the root of the document is considered to be. For fragment resolution methods (such as JSON Pointer fragments) that navigate through the document, the target of the "root" link should be the starting point for such methods.

The only exception is "root" links themselves. When calculating the target of links with relation "root", existing "root" links MUST NOT be taken into consideration.

For example, say we have the following schema:

```
{
    "links": [{
        "rel": "root",
        "href": "#/myRootData"
    }]
}
```

And the following data, returned from the URI:
"http://example.com/data/12345":

```
{
    "myRootData": {
        "title": "Document title"
    },
    "metaData": {
        ...
    }
}
```

To correctly resolve the URL "http://example.com/data/12345", we must take the "root" link into account. Here are some example URIs, along with the data they would resolve to:

URI	Data
http://example.com/data/12345	{"title": "Document title"}
http://example.com/data/12345#/title	"Document title"

5.2.2. 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:

```
Content-Type: application/json; profile=/schema-for-this-data

[{
    "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 value is advisory only, and is a schema that defines the expected structure of the JSON representation of the target of the link, if the target of the link is returned using JSON

representation.

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, you 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 the message text 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 2161, section 14.1 - HTTP "Accept" header [RFC2616].

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 2616, section 14.1 [RFC2616], 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 Link Properties

The following properties also apply to Link Description Objects, and provide functionality analogous to HTML forms, in providing a means for submitting extra (often user supplied) information to send to a server.

5.6.1. method

This property defines which method can be used to access the target resource. In an HTTP environment, this might be "GET" or "POST" (or other HTTP methods).

Some link relation values imply a set of appropriate HTTP methods to be used for the link. For example, a client might assume that a link with a relation of "edit" can be used in conjuction with the "PUT" HTTP method. If the client does not know which methods might be appropriate, then this SHOULD default to "GET".

5.6.2. encType

If present, this property indicates a query media type format that the server supports for querying or posting to the collection of instances at the target resource. The query can be suffixed to the target URI to query the collection with property-based constraints on the resources that SHOULD be returned from the server or used to post data to the resource (depending on the method).

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 no encType or method is specified, only the single URI specified by the href property is defined. 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 submitted request. For a GET request, this schema would define the properties for the query string and for a POST request, this would define the body.

Note that this is separate from the URI templating of "href" (which uses data from the instance, not submitted by the user). It is also separate from the "targetSchema" property, which provides a schema for the data that the client should expect to be returned when they follow the link.

6. IANA Considerations

6.1. Registry of Link Relations

This registry is maintained by IANA per RFC 4287 [RFC4287] and this specification adds four values: "full", "create", "instances", "root". New assignments are subject to IESG Approval, as outlined in

RFC 5226 [RFC5226]. Requests should be made by email to IANA, which will then forward the request to the IESG, requesting approval.

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, November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, December 2005.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, March 2012.

[json-pointer] Bryan, P., Zyp, K., and M. Nottingham, "JSON Pointer", August 2012, <http://tools.ietf.org/html/draft-ietf-appsawg-json-pointer-03>.

[json-schema-core] Galiegue, F., Zyp, K., and G. Court, "JSON Schema: core definitions and terminology", 2013, <http://tools.ietf.org/html/draft-zyp-json-schema-04>.

7.2. Informative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.

[W3C.REC-html401-19991224] Hors, A., Raggett, D., and I. Jacobs, "HTML 4.01 Specification", World Wide Web Consortium Recommendation REC-html401-19991224, December 1999, <http://www.w3.org/TR/1999/REC-html401-19991224>.

Appendix A. Change Log

draft-04

* Resolution of link URIs ("href") is now affected by rel="self" links on the instance
* Define "title" for LDOs
* Use URI Templates for the "href" property
* Split hyper-schema definition out from main schema.
* Capitalised the T in "encType", and the O in "readOnly"
* Moved "mediaType" and "contentEncoding" to the new "media" property (renamed "type" and "binaryEncoding")
* Added "mediaType" property to LDOs
* Replaced "slash-delimited" fragment resolution with "json-pointer".
* Added "template" LDO attribute.
* Improved wording of sections.

draft-03

* Added example and verbiage to "extends" attribute.

* Defined slash-delimited to use a leading slash.

* Made "root" a relation instead of an attribute.

* Removed address values, and MIME media type from format to reduce confusion (mediaType already exists, so it can be used for MIME types).

* Added more explanation of nullability.

* Removed "alternate" attribute.

* Upper cased many normative usages of must, may, and should.

* Replaced the link submission "properties" attribute to "schema" attribute.

* Replaced "optional" attribute with "required" attribute.

* Replaced "maximumCanEqual" attribute with "exclusiveMaximum" attribute.

* Replaced "minimumCanEqual" attribute with "exclusiveMinimum" attribute.

* Replaced "requires" attribute with "dependencies" attribute.

* Moved "contentEncoding" attribute to hyper schema.

* Added "additionalItems" attribute.

* Added "id" attribute.

* Switched self-referencing variable substitution from "-this" to "@" to align with reserved characters in URI template.

* Added "patternProperties" attribute.

* Schema URIs are now namespace versioned.

* Added "$ref" and "$schema" attributes.

draft-02

* Replaced "maxDecimal" attribute with "divisibleBy" attribute.

* Added slash-delimited fragment resolution protocol and made it the default.

* Added language about using links outside of schemas by referencing its normative URI.

* Added "uniqueItems" attribute.

* Added "targetSchema" attribute to link description object.

draft-01

* Fixed category and updates from template.

draft-00

* Initial draft.

Authors' Addresses

Geraint Luff (editor)
Cambridge
UK

EMail: luffgd@gmail.com

Kris Zyp
SitePen (USA)
530 Lytton Avenue
Palo Alto, CA 94301
USA

Phone: +1 650 968 8787
EMail: kris@sitepen.com

Gary Court
Calgary, AB
Canada

EMail: gary.court@gmail.com