Internet Engineering Task Force	K. Zyp, Ed.	
Internet-Draft	SitePen (USA)	
Intended status: Informational	December 05, 2009	
Expires: June 8, 2010		

TOC

A JSON Media Type for Describing the Structure and Meaning of JSON Documents
draft-zyp-json-schema-01

Abstract

JSON (JavaScript Object Notation) Schema defines the media type application/schema+json, a JSON based format for defining the structure of JSON data. JSON Schema provides a contract for what JSON data is required for a given application and how to interact with it. JSON Schema is intended to define validation, documentation, hyperlink navigation, and interaction control of JSON data.

Status of This Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.
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.”
The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.
The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.
This Internet-Draft will expire on June 8, 2010.

Copyright Notice

Table of Contents

---

1. Introduction

TOC

JSON (JavaScript Object Notation) Schema is a JSON media type for defining the structure of JSON data. JSON Schema provides a contract for what JSON data is required for a given application and how to interact with it. JSON Schema is intended to define validation, documentation, hyperlink navigation, and interaction control of JSON data.

---

2. Conventions

TOC

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.

---

3. Overview

TOC

JSON Schema defines the media type application/schema+json for describing the structure of other JSON documents. JSON Schema is JSON-based and includes facilities for describing the structure of JSON documents in terms of allowable values, descriptions, and interpreting relations with other resources.
JSON Schema format is organized into several separate definitions. The first definition is the core schema specification. This definition is primary concerned with describing a JSON structure and specifying valid elements in the structure. The second definition is the Hyper Schema specification which is intended define elements in a structure that can be interpreted as hyperlinks. Hyper Schema builds on JSON Schema to describe the hyperlink structure of other JSON documents. This allows

user agents to be able to successfully navigate JSON documents based on their schemas.
Cumulatively JSON Schema acts as a meta-document that can be used to define the required type and constraints on property values, as well as define the meaning of the property values for the purpose of describing a resource and determining hyperlinks within the representation.
An example JSON Schema that describes products might look like:

```
{
  "name":"Product",
  "properties":{
    "id":{
      "type":"number",
      "description":"Product identifier"
    },
    "name":{
      "description":"Name of the product",
      "type":"string"
    },
    "price":{
      "type": "number",
      "minimum":0
    },
    "tags":{
      "optional":true,
      "type":"array",
      "items":{
         "type":"string"
      }
    }
  },
  "links":[
    {
      "rel":"full",
      "href":"{id}"
    },
    {
      "rel":"comments",
      "href":"comments/?id={id}"
    }
  ]
}
```

This schema defines the properties of the instance JSON documents and their required properties (id, name, and price) as well as an optional property (tags). This also defines the link relations of the instance JSON documents.

3.1. Terminology

TOC

For this specification, a schema will be used to denote a JSON Schema definition, and an instance refers to the JSON object or array that the schema will be describing and validating

3.2. Design Considerations

TOC

The JSON Schema media type does not attempt to dictate the structure of JSON representations that contain data, but rather provides a separate format for flexibly communicating how a JSON representation should be interpreted and validated, such that user agents can properly understand acceptable structures and extrapolate hyperlink information with the JSON document. This specification does not define a protocol. The underlying protocol (such as HTTP) should sufficiently define the semantics of the client-server interface, the retrieval of resource representations linked to by JSON representations, and modification of those resources. The goal of this format is to sufficiently describe JSON structures such that one can utilize existing information available in existing JSON representations from a large variety of services that leverage a REST architecture using existing protocols.

4. Schema/Instance Association

TOC

JSON Schema instances are correlated to their schema by the "describedby" relation, where the schema is defined to be the target of the relation. Instance representations may be of the application/json media type or any other subtype. Consequently, dictating how an instance representation should specify the relation to the schema is beyond the normative scope of this document (since this document specifically defines the JSON Schema media type, and no other), but it is recommended that instances specify their schema so that user agents can interpret the instance representation and messages may retain the self-descriptive characteristic, avoiding the need for out-of-band information about instance data. Two approaches are recommended for declaring the relation to the schema that describes the meaning of a JSON instance's (or collection of instances) structure. A MIME type parameter named "describedby" or a Link header with a relation of "describedby" SHOULD be used:

```
Content-Type: application/json;
              describedby=http://json.com/my-hyper-schema
```

or if the content is being transferred by a protocol (such as HTTP) that provides headers, a Link header can be used:

```
Link: <http://json.com/my-hyper-schema>; rel="describedby"
```

Instances MAY specify multiple schemas, to indicate all the schemas that are applicable to the data. The instance data may have multiple schemas that it is defined by (the instance data should be valid for those schemas). Or if the document is a collection of instances, the collection may contain instances from different schemas. When collections contain heterogeneous instances, the pathStart attribute MAY be specified in the schema to disambiguate which schema should be applied for each item in the collection.

4.1. Self-Descriptive Schema

TOC

JSON Schemas are themselves instances for the schema schemas. A self-describing JSON Schema for the core JSON Schema can be found at http://json-schema.org/schema and the hyper schema self-description can be found at: http://json-schema.org/hyper-schema. All schemas used within a protocol with media type definitions SHOULD include a MIME parameter that refers to the self-descriptive hyper schema or another schema that extends this hyper schema:

```
Content-Type: application/json;
              describedby=http://www.json-schema.org/hyper-schema
```

---

5. Core Schema Definition

TOC

A JSON Schema is a JSON Object that defines various attributes of the instance and defines it's usage and valid values. A JSON Schema is a JSON Object with schema attribute properties. The following is the grammar of a JSON Schema:
And an example JSON Schema definition could look like:

```
{"description":"A person",
 "type":"object",

 "properties":
  {"name": {"type":"string"},
   "age" : {"type":"integer",
     "maximum":125}}
}
```

A JSON Schema object may have any of the following properties, called schema attributes (all attributes are optional):

5.1. type

TOC

> Union type definition - An array with two or more items which indicates a union of type definitions. Each item in the array may be a simple type definition or a schema. The instance value is valid if it is of the same type as one the type definitions in the array or if it is valid by one of the schemas in the array. For example to indicate that a string or number is a valid: {"type": ["string","number"]}
>
> Simple type definition - A string indicating a primitive or simple type. The following are acceptable strings:
>
> > string - Value must be a string.
> >
> > number - Value must be a number, floating point numbers are allowed.
> >
> > integer - Value must be an integer, no floating point numbers are allowed. This is a subset of the number type.
> >
> > boolean - Value must be a boolean.
> >
> > object - Value must be an object.
> >
> > array - Value must be an array.
> >
> > null - Value must be null. Note this is mainly for purpose of being able use union types to define nullability.
> >
> > any - Value may be of any type including null. If the property is not defined or is not in this list, than any type of value is acceptable. Other type values may be used for custom purposes,

> but minimal validators of the specification implementation can allow any instance value on unknown type values.

5.2. properties

TOC

This should be an object type definition, which is an object with property definitions that correspond to instance object properties. When the instance value is an object, the property values of the instance object must conform to the property definitions in this object. In this object, each property definition's value should be a schema, and the property's name should be the name of the instance property that it defines.

5.3. items

TOC

This should be a schema or an array of schemas. When this is an object/ schema and the instance value is an array, all the items in the array must conform to this schema. When this is an array of schemas and the instance value is an array, each position in the instance array must conform to the schema in the corresponding position for this array. This called tuple typing. When tuple typing is used, additional items are allowed, disallowed, or constrained by the additionalProperties attribute using the same rules as extra properties for objects..

5.4. optional

TOC

This indicates that the instance property in the instance object is optional. This is false by default.

5.5. additionalProperties

TOC

This provides a default property definition for all properties that are not explicitly defined in an object type definition. The value must be a schema. If false is provided, no additional properties are allowed, and the schema can not be extended. The default value is an empty schema which allows any value for additional properties.

5.6. requires

TOC

This indicates that if this property is present in the containing instance object, the property given by requires attribute must also be present in the containing instance object. The value of this property may be a string, indicating the require property name. Or the value may be a schema, in which case the containing instance must be valid by the schema if the property is present. For example if a object type definition is defined:

```
{
  "state":
  {
    "optional":true
  },
  "town":
  {
    "requires":"state",
    "optional":true
  }
}
```

An instance must include a state property if a town property is included. If a town property is not included, the state property is optional.

5.7. minimum

TOC

This indicates the minimum value for the instance property when the type of the instance value is a number.

5.8. maximum

TOC

This indicates the minimum value for the instance property when the type of the instance value is a number.

5.9. minimumCanEqual

TOC

If the minimum is defined, this indicates whether or not the instance property value can equal the minimum.

5.10. maximumCanEqual

TOC

If the maximum is defined, this indicates whether or not the instance property value can equal the maximum.

5.11. minItems

TOC

This indicates the minimum number of values in an array when an array is the instance value.

5.12. maxItems

TOC

This indicates the maximum number of values in an array when an array is the instance value.

5.13. pattern

TOC

When the instance value is a string, this provides a regular expression that a instance string value should match in order to be valid. Regular expressions should follow the regular expression specification from ECMA 262/Perl 5

5.14. maxLength

TOC

When the instance value is a string, this indicates maximum length of the string.

5.15. minLength

TOC

When the instance value is a string, this indicates minimum length of the string.

5.16. enum

TOC

This provides an enumeration of possible values that are valid for the instance property. This should be an array, and each item in the array represents a possible value for the instance value. If "enum" is included, the instance value must be one of the values in enum array in order for the schema to be valid.

5.17. title

TOC

This provides a short description of the instance property. The value must be a string.

5.18. description

TOC

This provides a full description of the of purpose the instance property. The value must be a string.

5.19. format

TOC

This property indicates the type of data, content type, or microformat to be expected in the instance property values. A format attribute may be one of the values listed below, and if so, should adhere to the semantics describing for the format. A format should only be used give meaning to primitive types (string, integer, number, or boolean). Validators are not required to validate that the instance values conform to a format. The following formats are defined:

> Any valid MIME media type may be used as a format value, in which case the instance property value must be a string, representing the contents of the MIME file.
>
> date-time - This should be a date in ISO 8601 format of YYYY-MM-DDThh:mm:ssZ in UTC time. This is the recommended form of date/ timestamp.
>
> date - This should be a date in the format of YYYY-MM-DD. It is recommended that you use the "date-time" format instead of "date" unless you need to transfer only the date part.

time - This should be a time in the format of hh:mm:ss. It is recommended that you use the "date-time" format instead of "time" unless you need to transfer only the time part.

utc-millisec - This should be the difference, measured in milliseconds, between the specified time and midnight, January 1, 1970 UTC. The value should be a number (integer or float).

regex - A regular expression.

color - This is a CSS color (like "#FF0000" or "red").

style - This is a CSS style definition (like "color: red; background-color:#FFF").

phone - This should be a phone number (format may follow E.123).

uri - This value should be a URI..

email - This should be an email address.

ip-address - This should be an ip version 4 address.

ipv6 - This should be an ip version 6 address.

street-address - This should be a street address.

locality - This should be a city or town.

region - This should be a region (a state in the US, province in Canada, etc.)

postal-code - This should be a postal code (AKA zip code).

country - This should be the name of a country.

Additional custom formats may be defined with a URL to a definition of the format.

5.20. contentEncoding

TOC

If the instance property value is a string, this indicates that the string should be interpreted as binary data and decoded using the encoding named by this schema property. RFC 2045, Sec 6.1 lists possible values.

5.21. default

TOC

This indicates the default for the instance property.

5.22. maxDecimal

TOC

This indicates the maximum number of decimal places in a floating point number. By default there is no maximum.

5.23. disallow

TOC

This attribute may take the same values as the "type" attribute, however if the instance matches the type or if this value is an array and the instance matches any type or schema in the array, than this instance is not valid.

5.24. extends

TOC

The value of this property should be another schema which will provide a base schema which the current schema will inherit from. The inheritance rules are such that any instance that is valid according to the current schema must be valid according to the referenced schema. This may also be an array, in which case, the instance must be valid for all the schemas in the array.

---

6. Hyper Schema

TOC

This section defines hypermedia definitions of JSON schema. The following attributes are specified in addition to those attributes that already provided by JSON schema with the specific purpose of informing user agents of relations between resources based on JSON data. Just as with JSON schema attributes, all the attributes in hyper-schema are optional. Therefore an empty object is a valid (non-informative) schema, and essentially describes plain JSON (no constraints on the structures). Addition of attributes provides additive information for user agents.

6.1. links

TOC

The value of the links property should be an array, where each item in the array is a link description object which describes the link relations of the instances.

6.1.1. Link Description Object

TOC

A link description object is used to describe the link relations of instances of a schema.

6.1.1.1. href

TOC

The value of the "href" link description property indicates the target URI of the related resource. The value of the instance property should be resolved as a URI-Reference per [RFC3986] and may be a relative URI. The base URI to be used for relative resolution should be the URI used to retrieve the instance object (not the schema). Also, the URI may be parametrized by the property values of the instance object.
Instance property values should be substituted into the URIs where matching braces ('{', '}') are found surrounding zero or more characters, creating an expanded URI. Instance property value substitutions are resolved by using the text between the braces to denote the property name from the instance to get the value to substitute. For example, if an href value is defined:

```
http://somesite.com/{id}
```

Then it would be resolved by replace the value of the "id" property value from the instance object. If the value of the "id" property was "45", the expanded URI would be:

```
http://somesite.com/45
```

If matching braces are found with the string "-this" (no quotes) between the braces, than the actual instance value should be used to replace the braces, rather than a property value. This should only be used in situations where the instance is a scalar (string, boolean, or number), and not for objects or arrays.

6.1.1.2. rel

TOC

The value of the "rel" property indicates the name of the relation to the target resource. 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. If a resource JSON representation contains a sub object with a property interpreted as a link, that sub-object holds the relation with the target. A relation to target from the top level resource 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 &rfc4287). However, we define these relation here for clarity of normative interpretation within the context of JSON hyper 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 object that contains this link may not be the full representation.
>
> describedby - This indicates the target of the link is the schema for the instance object. This may be used to specifically denote the schemas of objects within a JSON object hierarchy, facilitating polymorphic type data structures.
>
> The following relations are applicable for schemas (the schema as the "from" resource in the relation).
>
> instances - This indicates the target resource that represents 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 resource's JSON representation was retrieved:

```
GET /Resource/

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

This would indicate that for the first item in the collection, it's 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".

6.1.1.3. Submission Link Properties

TOC

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

6.1.1.3.1. method

TOC

This indicates which method should be used to access the target resource. In an HTTP environment, this would be "GET" or "POST" (other HTTP methods such as "PUT" and "DELETE" have semantics that are clearly implied by accessed resources, and do not need to be defined here). This defaults to "GET".

6.1.1.3.2. enctype

TOC

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:

```
/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.

TOC

6.1.1.3.3. properties

This is inherited from the base JSON schema definition, and can follow the same structure, but its meaning should be used to define the acceptable property names and values for the action (whether it be for the GET query or POST body). If properties are omitted, and this form is the child of a schema, the properties from the parent schema should be used as the basis for the form action.

6.2. fragmentResolution

TOC

This property indicates the fragment resolution protocol to use for resolving fragment identifiers in URIs within the instance representations. This applies to the instance object URIs and all children of the instance object's URIs. The default fragment resolution protocol is "dot-delimited", which is defined below. Other fragment resolution protocols may be used, but are not defined in this document. The fragment identifier is based on RFC 2396 Sec 5, and defines the mechanism for resolving references to entities within a document.

6.2.1. dot-delimited fragment resolution

TOC

With the dot-delimited fragment resolution protocol, the fragment identifier is interpreted as a series of property reference tokens that are delimited by the "." character (\x2E). Each property reference token is a series of any legal URI component characters except the "." character. Each property reference token should be interpreted, starting from the beginning of the fragment identifier, as a path reference in the target JSON structure. The final target value of the fragment can be determined by starting with the root of the JSON structure from the representation of the resource identified by the pre-fragment URI. If the target is a JSON object, than the new target is the value of the property with the name identified by the next property reference token in the fragment. If the target is a JSON array, than the target is determined by finding the item in array the array with the index defined by the next property reference token (which MUST be a number). The target is successively updated for each property reference token, until the entire fragment has been traversed. Property names SHOULD be URI-encoded. In particular, any "." in a property name MUST be encoded to avoid being interpreted as a property delimiter.
For example, for the following JSON representation:

```
{
   "foo":{
      "anArray":[
        {"prop":44}
      ],
      "another prop":{
          "baz":"A string"
      }
   }
}
```

The following fragment identifiers would be resolved:

fragment identifier	resolution
#	self, the root of the resource itself
#foo	the object referred to by the foo property
#foo.another prop	the object referred to by the "another prop" property of the object referred to by the "foo" property
#foo.another prop.baz	the string referred to by the value of "baz" property of the "another prop" property of the object referred to by the "foo" property
#foo.anArray.0	the first object in the "anArray" array

6.3. root

TOC

This attribute indicates that the value of the instance property value SHOULD be treated as the root or the body of the representation for the purposes of user agent interaction and fragment resolution (all other properties of the instance objects are can be regarded as meta-data descriptions for the data).

6.4. readonly

TOC

This indicates that the instance property should not be changed. Attempts by a user agent to modify the value of this property are expected to be rejected by a server.

6.5. pathStart

TOC

This property value is a URI-Reference that indicates the URI that all the URIs for the instances of the schema should start with. 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 URI of the instance begins with the pathStart's referenced URI. pathStart MUST be resolved as per [RFC3986] section 5. If the URI of the instance does not start with URI indicated by pathStart, or if another schema specifies a starting URI that is longer and also matches the instance, this schema should not be applied to the instance. Any schema that does not have a pathStart attribute should be considered applicable to all the instances for which it is referenced.

6.6. mediaType

TOC

This indicates the media type of the instance representations that this schema is defining.

6.7. alternate

TOC

This is an array of JSON schema definitions that define any other schemas for alternate JSON-based representations of the instance resources.

---

7. Security Considerations

TOC

This specification is a sub-type of the JSON format, and consequently the security considerations are generally the same as RFC 4627. However, an additional issue is that 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; describedby=/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"}
]
```

---

8. IANA Considerations

TOC

The proposed MIME media type for JSON Schema is application/schema+json
Type name: application
Subtype name: schema+json
Required parameters: describedby
The value of the describedby parameter should be a URI (relative or absolute) that refers to the schema used to define the structure of this structure (the meta-schema). Normally the value would be http://json-schema.org/hyper-schema, but it is allowable to use other schemas that extend the hyper schema's meta- schema.
Optional parameters: pretty

The value of the pretty parameter may be true or false to indicate if additional whitespace has been included to make the JSON representation easier to read.

8.1. Registry of Link Relations

TOC

This registry is maintained by IANA per RFC 4287 and this specification adds three values: "full", "create", "instances". New assignments are subject to IESG Approval, as outlined in [RFC5226]. Requests should be made by email to IANA, which will then forward the request to the IESG, requesting approval.

---

9. References

TOC

9.1. Normative References

TOC

[RFC3986]	Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” STD 66, RFC 3986, January 2005 (TXT, HTML, XML).
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC4287]	Nottingham, M., Ed. and R. Sayre, Ed., “The Atom Syndication Format,” RFC 4287, December 2005 (TXT, HTML, XML).
[RFC3339]	Klyne, G., Ed. and C. Newman, “Date and Time on the Internet: Timestamps,” RFC 3339, July 2002 (TXT, HTML, XML).
[RFC2045]	Freed, N. and N. Borenstein, “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies,” RFC 2045, November 1996 (TXT).

9.2. Informative References

TOC

[RFC4627]	Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON),” RFC 4627, July 2006 (TXT).
[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 (TXT, PS, PDF, HTML, XML).
[RFC5226]	Narten, T. and H. Alvestrand, “Guidelines for Writing an IANA Considerations Section in RFCs,” BCP 26, RFC 5226, May 2008 (TXT).
[I-D.hammer-discovery]	Hammer-Lahav, E., “LRDD: Link-based Resource Descriptor Discovery,” draft-hammer-discovery-04 (work in progress), March 2010 (TXT).
[I-D.gregorio-uritemplate]	Gregorio, J., Fielding, R., Hadley, M., and M. Nottingham, “URI Template,” draft-gregorio-uritemplate-04 (work in progress), March 2010 (TXT).
[I-D.nottingham-http-link-header]	Nottingham, M., “Web Linking,” draft-nottingham-http-link-header-09 (work in progress), April 2010 (TXT).
[W3C.REC-html401-19991224]	Hors, A., Jacobs, I., and D. Raggett, “HTML 4.01 Specification,” World Wide Web Consortium Recommendation REC-html401-19991224, December 1999 (HTML).

Appendix A. Change Log

TOC

-01

*Fixed category and updates from template

-00

*Initial draft

Appendix B. Open Issues

TOC

Should we give a preference to MIME headers over Link headers (or only use one)?

Should we use "profile" as the media type parameter instead?
Should "root" be a MIME parameter instead of a schema attribute?
Should "format" be renamed to "mediaType" or "contentType" to reflect the usage MIME media types that are allowed.
I still do not like how dates are handled.

Author's Address

TOC

	Kris Zyp (editor)
	SitePen (USA)
	530 Lytton Avenue
	Palo Alto, CA 94301
	USA
Phone:	+1 650 968 8787
EMail:	kris@sitepen.com