

Workgroup: HTTPAPI

Internet-Draft:

draft-ietf-httpapi-rest-api-mediatypes-03

Published: 12 March 2023

Intended Status: Informational

Expires: 13 September 2023

Authors: R. Polli

Digital Transformation Department, Italian Government

## **REST API Media Types**

### **Abstract**

This document registers the following media types used in APIs on the IANA Media Types registry: application/schema+json, application/schema-instance+json, application/openapi+json, and application/openapi+yaml.

### **About This Document**

This note is to be removed before publishing as an RFC.

Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-httpapi-rest-api-mediatypes/>.

Discussion of this document takes place on the HTTPAPI Working Group mailing list (<mailto:httpapi@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/httpapi/>. Subscribe at <https://www.ietf.org/mailman/listinfo/httpapi/>. Working Group information can be found at <https://datatracker.ietf.org/wg/httpapi/about/>.

Source for this draft and an issue tracker can be found at <https://github.com/ietf-wg-httpapi/mediatypes/labels/rest-api>.

### **Status of This Memo**

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

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

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

This Internet-Draft will expire on 13 September 2023.

## Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

- 1. [Introduction](#)
  - 1.1. [Notational Conventions](#)
- 2. [Media Type registrations](#)
  - 2.1. [The OpenAPI Media Types](#)
    - 2.1.1. [Media Type application/openapi+json](#)
    - 2.1.2. [Media Type application/openapi+yaml](#)
  - 2.2. [JSON Schema Media Types](#)
    - 2.2.1. [The "\\$schema" Keyword](#)
    - 2.2.2. [Identifying a Schema via a Media Type Parameter](#)
    - 2.2.3. [Linking to a Schema](#)
    - 2.2.4. [Fragment Identifiers](#)
    - 2.2.5. [Media Type application/schema+json](#)
    - 2.2.6. [Media Type application/schema-instance+json](#)
- 3. [Interoperability Considerations](#)
- 4. [Security Considerations](#)
- 5. [IANA Considerations](#)
- 6. [Normative References](#)
- [Appendix A. Acknowledgements](#)
- [FAQ](#)
- [Change Log](#)
  - [Since -00](#)
- [Author's Address](#)

## 1. Introduction

OpenAPI Specification [[oas](#)] version 3 and above is a consolidated standard for describing HTTP APIs using the JSON [[JSON](#)] and YAML [[YAML](#)] data format.

YAML media type registration is addressed in [[YAML-MEDIATYPES](#)], which provides interoperability and security considerations.

To increase interoperability when processing API specifications and leverage content negotiation mechanisms when exchanging OpenAPI Specification resources this specification register the following media-types: application/schema+json, application/schema-instance+json, application/openapi+json and application/openapi+yaml.

### 1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

This document uses the Augmented BNF defined in [[RFC5234](#)] and updated by [[RFC7405](#)].

The terms "content", "content negotiation", "resource", and "user agent" in this document are to be interpreted as in [[SEMANTICS](#)].

## 2. Media Type registrations

This section describes the information required to register the above media types according to [[MEDIATYPE](#)]

### 2.1. The OpenAPI Media Types

The OpenAPI Specification Media Types convey OpenAPI document (OAS) files as defined in [[oas](#)] for version 3.0.0 and above.

Those files can be serialized in [[JSON](#)] or [[YAML](#)]. Since there are multiple OpenAPI Specification versions, those media-types support the version parameter.

The following examples conveys the desire of a client to receive an OpenAPI Specification resource preferably in the following order:

1. openapi 3.1 in YAML
2. openapi 3.0 in YAML
3. any openapi version in json

Accept: application/openapi+yaml;version=3.1,  
application/openapi+yaml;version=3.0;q=0.5,  
application/openapi+json;q=0.3

### 2.1.1.1. Media Type application/openapi+json

The following information serves as the registration form for the application/openapi+json media type.

**Type name:** application

**Subtype name:** openapi+json

**Required parameters:** None

**Optional parameters:** version; unrecognized parameters should be ignored

**Encoding considerations:** Same as [[JSON](#)]

**Security considerations:** see [Section 4](#) of this document

**Interoperability considerations:** None

**Published specification:** this document

**Applications that use this media type:** HTTP

**Fragment identifier considerations:** Same as for application/json [[JSON](#)]

Additional information:

\*Deprecated alias names for this type: n/a

\*Magic number(s): n/a

\*File extension(s): json

\*Macintosh file type code(s): n/a

**Person and email address to contact for further information:** See Authors' Addresses section.

**Intended usage:** COMMON

**Restrictions on usage:** None.

**Author:** See Authors' Addresses section.

**Change controller:** n/a

### 2.1.2. Media Type application/openapi+yaml

The following information serves as the registration form for the application/openapi+yaml media type.

**Type name:** application

**Subtype name:** openapi+yaml

**Required parameters:** None

**Optional parameters:** version; unrecognized parameters should be ignored

**Encoding considerations:** See [[YAML-MEDIATYPES](#)]

**Security considerations:** see [Section 4](#) of [[YAML-MEDIATYPES](#)]

**Interoperability considerations:** see [Section 3](#) of [[YAML-MEDIATYPES](#)]

**Published specification:** this document

**Applications that use this media type:** HTTP

**Fragment identifier considerations:** Same as for application/json [[JSON](#)]

**Additional information:**

\*Deprecated alias names for this type: n/a

\*Magic number(s): n/a

\*File extension(s): yaml, yml

\*Macintosh file type code(s): n/a

**Person and email address to contact for further information:** See Authors' Addresses section

**Intended usage:** COMMON

**Restrictions on usage:** None.

**Author:** See Authors' Addresses section

**Change controller:** n/a

## 2.2. JSON Schema Media Types

JSON Schema is a declarative domain-specific language for validating and annotating JSON documents (see [[jsonschema](#)]).

This document registers the media types associated with JSON Schema.

There are many dialects of JSON Schema in wide use today. The JSON Schema maintainers have released several dialects including draft-04, draft-07, and draft 2020-12. There are also several third-party JSON Schema dialects in wide use including the ones defined for use in OpenAPI and MongoDB.

This specification defines little more than how to identify the dialect while leaving most of the semantics of the schema up to the dialect to define. Clients MUST use the following order of precedence for determining the dialect of a schema.

- \*The `$schema` keyword ([Section 2.2.1](#))
- \*The "schema" media type parameter ([Section 2.2.2](#))
- \*The context of the enclosing document. This applies only when a schema is embedded within a document. The enclosing document could be another schema in the case of a bundled schema or it could be another type of document that includes schemas such as an OpenAPI document.
- \*If none of the above result in identifying the dialect, client behavior is undefined.

### 2.2.1. The "\$schema" Keyword

The `$schema` keyword is used as a JSON Schema dialect identifier. The value of this keyword MUST be a URI [[RFC3986](#)]. This URI SHOULD identify a meta-schema that can be used to validate that the schema is syntactically correct according to the dialect the URI identifies.

The dialect SHOULD define where the `$schema` keyword is allowed and/or recognized in a schema, but it is RECOMMENDED that dialects do not allow the schema to change within the same Schema Resource.

### 2.2.2. Identifying a Schema via a Media Type Parameter

Media types MAY allow for a schema media type parameter, to support content negotiation based on schema identifier (see [Section 12](#) of [[SEMANTICS](#)]). The schema media type parameter MUST be a URI-reference [[RFC3986](#)].

The schema parameter identifies a schema that provides semantic information about the resource the media type represents. When using the application/schema+json media type, the schema parameter identifies the dialect of the schema the media type represents.

The schema URI is opaque and SHOULD NOT automatically be dereferenced. Since schema doesn't necessarily point to a network location, the "describedby" relation is used for linking to a downloadable schema.

The following is an example of content negotiation where a user agent can accept two different versions of a "pet" resource. Each resource version is identified by a unique JSON Schema.

Request:

NOTE: '\' line wrapping per RFC 8792

```
GET /pet/1234 HTTP/1.1
Host: foo.example
Accept: \
  application/schema-instance+json; schema="/schemas/v2/pet"; q=0.2, \
  application/schema-instance+json; schema="/schemas/v1/pet"; q=0.1
```

Response:

NOTE: '\' line wrapping per RFC 8792

```
HTTP/1.1 200 Ok
Content-Type: \
  application/schema-instance+json; schema="/schemas/v2/pet"

{
  "petId": "1234",
  "name": "Pluto",
  ...
}
```

In the following example, the user agent is able to accept two possible dialects of JSON Schema and the server replies with the latest one.

Request:

NOTE: '\\' line wrapping per RFC 8792

```
GET /schemas/v2/pet HTTP/1.1
Host: foo.example
Accept: application/schema+json; \
      schema="https://json-schema.org/draft/2020-12/schema", \
      application/schema+json; \
      schema="http://json-schema.org/draft-07/schema#"
```

Response:

NOTE: '\\' line wrapping per RFC 8792

```
HTTP/1.1 200 OK
Content-Type: \
  application/schema+json; \
  schema="https://json-schema.org/draft/2020-12/schema"

{
  "$id": "https://json-schema.org/draft/2020-12/schema",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  ...
}
```

### 2.2.3. Linking to a Schema

It is RECOMMENDED that instances described by a schema provide a link to a downloadable JSON Schema using the link relation described by, as defined by Linked Data Protocol 1.0, section 8.1 [[W3C.REC-ldp-20150226](#)].

In HTTP, such links can be attached to any response using the Link header [[LINK](#)].

Link: <<https://example.com/my-hyper-schema#>>; rel="describedby"

### 2.2.4. Fragment Identifiers

Two fragment identifier structures are supported: JSON Pointers and plain-names.

The use of JSON Pointers as URI fragment identifiers is described in [[JSON-POINTER](#)]. Fragment identifiers that are empty or start with a /, MUST be interpreted as JSON Pointer fragment identifiers.

Plain-name fragment identifiers reference locally named locations in the document. The dialect determines how plain-name identifiers map to locations within the document. All fragment identifiers that do not match the JSON Pointer syntax MUST be interpreted as plain name fragment identifiers.



### 2.2.5. Media Type application/schema+json

The application/schema+json media type represents JSON Schema. This schema can be an official dialect or a third-party dialect. The following information serves as the registration form for the application/schema+json media type.

**Type name:** application

**Subtype name:** schema+json

**Required parameters:** N/A

**Optional parameters:**

\*schema: A URI identifying the JSON Schema dialect the schema was written for. If this value conflicts with the value of the \$schema keyword in the schema, the \$schema keyword takes precedence.

**Encoding considerations:** Same as [\[JSON\]](#)

**Security considerations:** See the "Security Considerations" section of [\[jsonschema\]](#)

**Interoperability considerations:** See the "General Considerations" section of [\[jsonschema\]](#)

**Published specification:** this document

**Applications that use this media type:** JSON Schema is used in a variety of applications including API servers and clients that validate JSON requests and responses, IDEs that valid configuration files, databases that store JSON, and more.

**Fragment identifier considerations:** See [Section 2.2.4](#)

**Additional information:**

\*Deprecated alias names for this type: N/A

\*Magic number(s): N/A

\*File extension(s): json, schema.json

\*Macintosh file type code(s): N/A

**Person and email address to contact for further information:** See Authors' Addresses section.

**Intended usage:**

COMMON

**Restrictions on usage:** N/A.

**Author:** See Authors' Addresses section.

**Change controller:** N/A

**2.2.6. Media Type application/schema-instance+json**

The application/schema-instance+json media type is an extension of the [[JSON](#)] media type that just adds the schema media type parameter and fragment identification. The following information serves as the registration form for the application/schema-instance+json media type.

**Type name:** application

**Subtype name:** schema-instance+json

**Required parameters:** N/A

Optional parameters:

\***schema:** A URI identifying a JSON Schema that provides semantic information about this JSON representation.

**Encoding considerations:** Same as [[JSON](#)]

**Security considerations:** Same as [[JSON](#)]

**Interoperability considerations:** Same as [[JSON](#)]

**Published specification:** this document

**Applications that use this media type:** JSON Schema is used in a variety of applications including API servers and clients that validate JSON requests and responses, IDEs that valid configuration files, databases that store JSON, and more.

**Fragment identifier considerations:** See [Section 2.2.4](#)

Additional information:

\*Deprecated alias names for this type: N/A

\*Magic number(s): N/A

\*File extension(s): json

\*Macintosh file type code(s): N/A

**Person and email address to contact for further information:** See Authors' Addresses section.

**Intended usage:** COMMON

**Restrictions on usage:** N/A

**Author:** See Authors' Addresses section.

**Change controller:** N/A

### 3. Interoperability Considerations

Interoperability requirements for media type registrations are discussed in Section 4.6 of [[MEDIATYPE](#)]. and [Section 3](#) of [[YAML-MEDIATYPES](#)].

### 4. Security Considerations

Security requirements for media type registrations are discussed in Section 4.6 of [[MEDIATYPE](#)]. and [Section 4](#) of [[YAML-MEDIATYPES](#)].

### 5. IANA Considerations

This specification defines the following new Internet media types [[MEDIATYPE](#)].

IANA has updated the "Media Types" registry at <https://www.iana.org/assignments/media-types> with the registration information provided below.

Media Type	Section
application/openapi+yaml	<a href="#">Section 2.1.2</a> of this document
application/openapi+json	<a href="#">Section 2.1.1</a> of this document
application/schema+json	<a href="#">Section 2.2.5</a> of this document
application/schema-instance+json	<a href="#">Section 2.2.6</a> of this document

Table 1

### 6. Normative References

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

**[JSON-POINTER]** Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901,

DOI 10.17487/RFC6901, April 2013, <<https://www.rfc-editor.org/rfc/rfc6901>>.

[**jsonschema**] Wright, A., Andrews, H., Hutton, B., and G. Dennis, "JSON Schema Core", 28 January 2020, <<https://json-schema.org/specification.html>>.

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

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

[**oas**] Darrel Miller, Jeremy Whitlock, Marsh Gardiner, Mike Ralphson, Ron Ratovsky, and Uri Sarid, "OpenAPI Specification 3.0.0", 26 July 2017.

[**RFC2119**] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/rfc/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, <<https://www.rfc-editor.org/rfc/rfc3986>>.

[**RFC5234**] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/rfc/rfc5234>>.

[**RFC7405**] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <<https://www.rfc-editor.org/rfc/rfc7405>>.

[**RFC8174**] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.

[**SEMANTICS**] Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantics-19, 12 September 2021, <<https://>>

[datatracker.ietf.org/doc/html/draft-ietf-httpbis-semantics-19](https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-semantics-19)>.

[W3C.REC-ldp-20150226] Malhotra, A., Ed., Arwe, J., Ed., and S. Speicher, Ed., "Linked Data Platform 1.0", W3C REC REC-ldp-20150226, W3C REC-ldp-20150226, 26 February 2015, <<https://www.w3.org/TR/2015/REC-ldp-20150226/>>.

[YAML] Oren Ben-Kiki, Clark Evans, and Ingy dot Net, "YAML Ain't Markup Language Version 1.2", 1 October 2021, <<https://yaml.org/spec/1.2/spec.html>>.

[YAML-MEDIATYPES] Polli, R., Wilde, E., and E. Aro, "YAML Media Type", Work in Progress, Internet-Draft, draft-ietf-httpapi-yaml-mediatypes-04, 23 November 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-httpapi-yaml-mediatypes-04>>.

## Appendix A. Acknowledgements

Thanks to Erik Wilde and David Biesack for being the initial contributors of this specification, and to Darrel Miller and Rich Salz for their support during the adoption phase.

In addition to the people above, this document owes a lot to the extensive discussion inside and outside the HTTPAPI workgroup. The following contributors have helped improve this specification by opening pull requests, reporting bugs, asking smart questions, drafting or reviewing text, and evaluating open issues:

Eemeli Aro, Tina (tinita) Müller, Ben Hutton and Jason Desrosiers.

## FAQ

**Q: Why this document?** After all these years, we still lack a proper media-type for REST related document types. This has some security implications too (eg. wrt on identifying parsers or treat downloads)

## Change Log

RFC EDITOR PLEASE DELETE THIS SECTION.

## Since -00

\*Split YAML registrations in a separate I-D.

## Author's Address

Roberto Polli

Digital Transformation Department, Italian Government  
Italy

Email: [robipolli@gmail.com](mailto:robipolli@gmail.com)