

HTTPAPI
Internet-Draft Digital Transformation Department, Italian Government
Intended status: Informational
Expires: 3 October 2022

R. Polli
E. Wilde
Axway
E. Aro
1 April 2022

YAML Media Types
draft-ietf-httpapi-yaml-mediatypes-00

Abstract

This document registers the application/yaml media type and the +yaml structured syntax suffix on the IANA Media Types registry.

Note to Readers

RFC EDITOR: please remove this section before publication

Discussion of this draft takes place on the HTTP APIs working group mailing list (httpapi@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/browse/httpapi/> (<https://mailarchive.ietf.org/arch/browse/httpapi/>).

The source code and issues list for this draft can be found at <https://github.com/ietf-wg-httpapi/mediatypes> (<https://github.com/ietf-wg-httpapi/mediatypes>).

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 3 October 2022.

Internet-Draft

YAML Media Types

April 2022

Copyright Notice

Copyright (c) 2022 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	2
1.1.	Notational Conventions	3
2.	Media Type registrations	3
2.1.	Media Type application/yaml	3
2.2.	The +yaml Structured Syntax Suffix	4
3.	Interoperability Considerations	5
3.1.	YAML is an Evolving Language	5
3.2.	YAML and JSON	5
4.	Security Considerations	6
4.1.	Arbitrary Code Execution	6
4.2.	Resource Exhaustion	7
5.	IANA Considerations	7
6.	Normative References	8
Appendix A.	Acknowledgements	9
	FAQ	9
	Change Log	9
	Authors' Addresses	9

[1.](#) Introduction

YAML [[YAML](#)] is a data serialization format that is widely used on the Internet, including in the API sector (e.g. see [[oas](#)]) but the relevant media type and structured syntax suffix are not registered by IANA.

To increase interoperability when exchanging YAML data and leverage content negotiation mechanisms when exchanging YAML resources, this

specification registers the application/yaml media type and the +yaml structured syntax suffix.

Moreover, it provides security considerations and interoperability considerations related to [\[YAML\]](#), including its relation with [\[JSON\]](#).

[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.](#) Media Type application/yaml

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

Type name: application

Subtype name: yaml

Required parameters: None

Optional parameters: None; unrecognized parameters should be ignored

Encoding considerations: binary

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

Interoperability considerations: see [Section 3](#) of this document

Published specification: this document

Applications that use this media type: HTTP

Fragment identifier considerations: None

Additional information:

Polli, et al.

Expires 3 October 2022

[Page 3]

Internet-Draft

YAML Media Types

April 2022

- * Deprecated alias names for this type: application/x-yaml, text/yaml, text/x-yaml
- * 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.](#) The +yaml Structured Syntax Suffix

The suffix +yaml MAY be used with any media type whose representation follows that established for application/yaml. The media type structured syntax suffix registration form follows. See [[MEDIATYPE](#)] for definitions of each of the registration form headings.

Name: YAML Ain't Markup LanguageML (YAML)

+suffix: +yaml

References: [\[YAML\]](#)

Encoding considerations: see [Section 2.1](#)

Fragment identifier considerations: The syntax and semantics of fragment identifiers specified for +yaml SHOULD be as specified for [Section 2.1](#). The syntax and semantics for fragment identifiers for a specific xxx/yyy+yaml SHOULD be processed as follows:

1. For cases defined in +yaml, where the fragment identifier resolves per the +yaml rules, then process as specified in +yaml.
2. For cases defined in +yaml, where the fragment identifier does not resolve per the +yaml rules, then process as specified in xxx/yyy+yaml.

3. For cases not defined in +yaml, then process as specified in xxx/yyy+yaml.

Interoperability considerations: See [Section 2.1](#)

Security considerations: See [Section 2.1](#)

Contact: See Authors' Addresses section.

Author: See Authors' Addresses section

Change controller: n/a

[3.](#) Interoperability Considerations

[3.1.](#) YAML is an Evolving Language

YAML is an evolving language and, in time, some features have been added, and others removed.

While this document is based on a given YAML version [\[YAML\]](#), media types registration does not imply a specific version. This allows

content negotiation of version-independent YAML resources.

Implementers concerned about features related to a specific YAML version can specify it in the documents using the %YAML directive (see Section 6.8.1 of [\[YAML\]](#)).

[3.2.](#) YAML and JSON

When using flow collection styles (see Section 7.4 of [\[YAML\]](#)) a YAML document could look like JSON [\[JSON\]](#), thus similar interoperability considerations apply.

When using YAML as a more efficient format to serialize information intended to be consumed as JSON, information can be discarded: this includes comments (see Section 3.2.3.3 of [\[YAML\]](#)) and alias nodes (see Section 7.1 of [\[YAML\]](#)), that do not have a JSON counterpart.

```
# This comment will be lost
# when serializing in JSON.
Title:
  type: string
  maxLength: &text_limit 64

Name:
  type: string
  maxLength: *text_limit # Replaced by the value 64.
```

Figure 1: JSON replaces alias nodes with static values.

Implementers need to ensure that relevant information will not be lost during the processing. For example, they might consider acceptable that alias nodes are replaced by static values.

In some cases an implementer may want to define a list of allowed YAML features, taking into account that the following ones might have interoperability issues with JSON:

- * non UTF-8 encoding, since YAML supports UTF-16 and UTF-32 in addition to UTF-8;
- * mapping keys that are not strings;

- * circular references represented using anchor (see [Section 4.2](#) and Figure 3);
- * `.inf` and `.nan` float values, since JSON does not support them;
- * non-JSON types, including the ones associated with tags like `!!timestamp` that were included in the default schema of older YAML versions;
- * tags in general, and specifically the ones that do not map to JSON types like custom and local tags such as `!!python/object` and `!mytag` (see Section 2.4 of [\[YAML\]](#));

```

non-json-keys:
  2020-01-01: a timestamp
  [0, 1]: a sequence
  ? {k: v}
    : a map
non-json-value: 2020-01-01

```

Figure 2: Example of mapping keys not supported in JSON

[4.](#) Security Considerations

Security requirements for both media type and media type suffix registrations are discussed in Section 4.6 of [\[MEDIATYPE\]](#).

[4.1.](#) Arbitrary Code Execution

Care should be used when using YAML tags, because their resolution might trigger unexpected code execution.

Code execution in deserializers should be disabled by default, and only be enabled explicitly. In those cases, the implementation should ensure – for example, via specific functions – that the code execution results in strictly bounded time/memory limits.

Many implementations provide safe deserializers addressing these issues.

4.2. Resource Exhaustion

YAML documents are rooted, connected, directed graphs and can contain reference cycles, so they can't be treated as simple trees (see Section 3.2.1 of [YAML]). An implementation that attempts to do that can infinite-loop at some point (e.g. when trying to serialize a YAML document in JSON).

```
x: &x
  y: *x
```

Figure 3: A cyclic document

Even if a document is not cyclic, treating it as a simple tree could lead to improper behaviors (such as the "billion laughs" problem).

```
x1: &a1 ["a", "a"]
x2: &a2 [*a1, *a1]
x3: &a3 [*a2, *a2]
```

Figure 4: A billion laughs document

This can be addressed using processors limiting the anchor recursion depth and validating the input before processing it; even in these cases it is important to carefully test the implementation you are going to use. The same considerations apply when serializing a YAML representation graph in a format that does not support reference cycles (see [Section 3.2](#)).

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> (<https://www.iana.org/assignments/media-types>) with the registration information provided below.

Media Type	Section	
=====+	=====+	=====+
application/yaml	Section 2.1 of this document	
+-----+	+-----+	+-----+

Table 1

IANA has updated the "Structured Syntax Suffixes" registry at <https://www.iana.org/assignments/media-type-structured-suffix> (<https://www.iana.org/assignments/media-type-structured-suffix>) with the registration information provided below.

=====+	=====+	=====+
Suffix	Section	
=====+	=====+	=====+
+yaml	Section 2.2 of this document	
+-----+	+-----+	+-----+

Table 2

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>>.
- [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>>.
- [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>>.
- [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>>.

[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:

Tina (tinita) Mueller, Ben Hutton, Manu Sporny and Jason Desrosiers.

FAQ

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

Change Log

RFC EDITOR PLEASE DELETE THIS SECTION.

Authors' Addresses

Roberto Polli

Email: robipolli@gmail.com

Erik Wilde
Axway
Switzerland
Email: erik.wilde@dret.net

Eemeli Aro
Finland
Email: eemeli@gmail.com

