Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 17, 2019

A. Rundgren
Independent

February 13, 2019

"Comparable" JSON (JSONCOMP)

draft-rundgren-comparable-json-04

Abstract

This application note describes how JCS [JCS] can be utilized to support applications needing canonicalization beyond the core JSON [RFC8259] level, with comparisons as the primary target.

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 August 17, 2019.

Copyright Notice

Table of Contents

1. Introduction

The purpose of JCS [JCS] is creating "Hashable" representations of JSON [RFC8259] data intended for cryptographic solutions. JCS accomplishes this by combining normalization of the native JSON String and Number primitives with a deterministic property sorting scheme. That is, JCS provides canonicalization at the core JSON level. For interoperability reasons JCS also constrains data to the I-JSON [RFC7493] subset.

However, if you rather would like to compare JSON data from different sources or runs, JCS would in many cases be inadequate since the JSON String type is commonly used for holding subtypes like "DateTime" or "BigInteger" objects.

This application note outlines how JCS in spite of having a limited canonicalization scope still may be utilized by applications like above.

2. Terminology

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.

3. String Subtype Normalization

Assume you want to compare productions of JSON data where the schema designer assigned the property "big" for holding a "BigInteger" subtype and "time" for holding a "DateTime" subtype, while "val" is supposed to be a JSON Number compliant with JCS. The following example shows such an object:

```
{
  "time": "2019-01-28T07:45:10Z",
  "big": "055",
  "val": 3.5
}
```

A problem here is that "055" clearly is not a canonical form for a "BigInteger" while a "DateTime" object like "2019-01-28T07:45:10Z" might as well be expressed as "2019-01-28T08:45:10.000+01:00" making comparisons based on JCS canonicalization fail.

To resolve this issue using JCS the following measures MUST be taken:

- The community or standard utilizing a specific JSON schema defines a strict normalized form for each of the used subtypes.

- Compatible serializers are created for each subtype.

A positive side effect of this arrangement is that it enforces strict definitions of subtypes which improves interoperability in general as well.

Defining specific subtypes and their normalized form is out of scope for this application note. Although the JSON example illustrated a "BigInteger" in decimal notation, applications transferring huge integers (like raw RSA keys) typically rather use Base64 [RFC4648] encoding to conserve space.

Below is an example of a strict serializer expressed in ECMAScript [ECMASCRIPT] for a "DateTime" subtype:

```
Date.prototype.toJSON = function() {
    let date = this.toISOString();
    // In this particular case an ISO/UTC notation was selected
    // yyyy-mm-ddThh:mm:ssZ
    return date.substring(0, date.indexOf('.')) + 'Z';
};
```

4. IANA Considerations

This document has no IANA actions.

5. Security Considerations

Systems implementing this application note are subject to the same security considerations as JCS.

6. Acknowledgements

This document was created based on feedback (on JCS) from many people including Mark Nottingham and Jim Schaad.

7. References

7.1. Normative References

[JCS] A. Rundgren, B. Jordan, S. Erdtman, "JSON Canonicalization Scheme - Work in progress", <https://tools.ietf.org/html/draft-rundgren-json-canonicalization-scheme-05>.

[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/info/rfc2119>.

[RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <https://www.rfc-editor.org/info/rfc7493>.

[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/info/rfc8174>.

[RFC8259] 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/info/rfc8259>.

7.2. Informal References

[ECMASCRIPT] Ecma International, "ECMAScript 2015 Language Specification", <https://www.ecma-international.org/ecma-262/6.0/index.html>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

Author's Address

Anders Rundgren
Independent
Montpellier
France

Email: anders.rundgren.net@gmail.com
URI: https://www.linkedin.com/in/andersrundgren/