Internet-Draft
Intended status: Standards Track
Expires: May 07, 2013

G. Staykov
VMware
J. Hu
VMware

November 07, 2012

JSON Canonical Form
draft-staykov-hu-json-canonical-form-00

Abstract

A single JSON document can have multiple logically equivalent physical representations. While convenient for human interaction, this flexibility is inconvenient for cases where a machine is used to assess the logical equivalence of documents. In cases where logical equivalence is useful, an encoder should produce a canonical form of a JSON document. For example, since digital signatures demand the same physical representation for logically equivalent documents, a canonical physical representation would allow the signature to apply to the logical document. This internet draft has the goal to define a canonical form of JSON documents. Two logically equivalent documents should have same canonical form.

Requirements Language

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 [RFC2119].

Status of this Memo

Copyright Notice

1. Introduction

JSON [JSON] is a lightweight data-interchange text format that is suitable for both humans and machines. It allows multiple physical representations that are logically equivalent. For example, a formatting change to add whitespaces and line endings to make a document more human readable will result in a different representation when doing a byte for byte comparison. There are cases however where it is essential to have a single physical representation of a data document. For example when a cryptographic hash is applied over a JSON document, a single physical representation allows the hash to represent the logical content of the document by removing variation in how that content is encoded in JSON. Thus a common physical representation of logically equivalent JSON documents should be defined. It is called canonical form.

2. JSON canonical form

The canonical form is defined by the following rules:

* The document MUST be encoded in UTF-8 [UTF-8]
* Non-significant(1) whitespace characters MUST NOT be used
* Non-significant(1) line endings MUST NOT be used
* Entries (set of name/value pairs) in JSON objects MUST be sorted lexicographically(2) by their names
* Arrays MUST preserve their initial ordering

(1)As defined in JSON data-interchange format [JSON], JSON objects consists of multiple "name"/"value" pairs and JSON arrays consists of multiple "value" fields. Non-significant means not part of "name" or "value".

(2)Lexicographic comparison, which orders strings from least to greatest alphabetically based on the UCS (Unicode Character Set) codepoint values.

2.1 Canonical representation of data types

2.1.1 Double

The double data type is represented as specified in the XML schema standard [XML]

* The canonical representation of the double data type consists of mantissa followed by "E", followed by exponent.
* Mantissa
  * MUST be represented as a decimal. The decimal point is mandatory
  * There MUST be a single non zero digit on the left of the decimal point (unless a zero is represented).
  * There MUST be at least single digit on the right of the decimal point.
* Exponent
  * Zero exponent is represented by "E0".
* "+" sign is prohibited in both the mantissa and the exponent.
* Leading zeroes are prohibited from the left side of the decimal point in the mantissa and from the exponent.
* Special values (NaN, INF) MUST not be used.

3. Applications

The JSON canonical form can be used when digitally signing JSON documents generated from a serialization library. Because serialization and deserialization libraries might tolerate variation in physical representation, different physical representations may result after several serialization / deserialization cycles. This could result in false signature verification failures as the hash digest of the same document differs from the hash digest used when signing. A way to avoid this problem is to use canonical form when signing and verifying hash digests.

4. Examples

4.1. Example 1

Input:

```
{
  "foo" : "foo bar"
}
```

Canonical form:

```
{"foo":"foo bar"}
```

Demonstrates:

* Non-significant whitespace characters and line endings are removed.
* Whitespaces inside name/value object entities are preserved.

4.2. Example 2

Input:

```
{
  "foo":"bar",
  "abc":"def",
```

```
    "zoo" :
      [
        "def",
        "abc"
      ]
  }
```

```
  Canonical Form:
  {"abc":"def","foo":"bar","zoo":["def","abc"]}
```

```
  Demonstrates:
  *  Non-significant whitespaces and line endings are removed.
  *  Name/value pairs in JSON objects are lexicographically sorted by
   "name" key.
  *  Array order is preserved.
```

4.3. Example 3

```
  Input:
  {
    "d1":-12.34e4,
    "d2":1E-130,
    "d3":0.0E-0,
    "d4":1.2
  }
```

```
  Canonical Form:
  {"d1":-1.234E5,"d2":1.0E-130,"d3":0.0E0,"d4":1.2E0}
```

```
  Demonstrates:
  *  Various canonical representations of double data types.
```

5. Security Considerations

This document provides a groundwork needed for providing data integrity by using digital signatures over JSON messages.

6. IANA Considerations

This document has no actions for IANA

7. References

7.1. Normative References

[JSON] http://www.json.org/

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

[UTF-8] UTF-8, a transformation format of ISO 10646, IETF RFC 3629.

F. Yergeau. January 1998.
http://www.ietf.org/rfc/rfc3629.txt

[XML] http://www.w3.org/TR/xmlschema-2

Authors' Addresses

Georgi Staykov
VMware
Email: gstaykov@vmware.com

Jeff Hu
VMware
Email: jhu@vmware.com